Towards High Efficient Federated Learning Systems

JIAMING PEI

Doctor of Philosophy

Supervisor: Dr. Wei Li
Associate Supervisor: Prof. Albert Y. Zomaya

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

17 March 2026

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Wei Li, and Professor Albert Y. Zomaya, for their supervision, continuous support, and insightful feedback throughout the course of my research. Their expertise and patience have been instrumental in shaping both this thesis and my academic growth. I sincerely appreciate the time and effort they dedicated to discussing ideas, reviewing drafts, and encouraging me during difficult moments. It has been a privilege to learn under their mentorship. This research reported in this thesis was supported by the award of a Research Training Program scholarship to me.

I am also sincerely thankful to my friends and peers who accompanied me through the highs and lows of this journey. Their companionship, discussions, and moral support made the research process not only manageable but also enjoyable.

Finally, I owe my heartfelt thanks to my family for their unwavering love, understanding, and support. Their belief in me provided the strength I needed to persevere and complete this thesis.

Without the support of all these people, this work would not have been possible.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

I certify that this thesis does not use any generative artificial intelligence (AI) tool.

Name: Jiaming Pei **Signature:**

Date: 17th March 2026

Student Plagiarism: Compliance Statement

I certify that:

I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

This Work is substantially my own, and to the extent that any part of this Work is not my own I have indicated that it is not my own by Acknowledging the Source of that part or those parts of the Work.

Name: Jiaming Pei

Date: 17th March 2026

Abstract

Federated learning (FL) has shown strong potential in domains such as Intelligent Transportation Systems (ITS), Internet of Things (IoT), and industrial cyber–physical systems (ICPS). However, FL faces three fundamental challenges: communication heterogeneity, statistical heterogeneity, and system heterogeneity. Communication heterogeneity arises from differences in network conditions and transmission capacity among clients, statistical heterogeneity results from non-IID data distributions across clients, and system heterogeneity reflects variations in computational resources and client availability. Existing studies often address these challenges independently, leaving their interactions insufficiently understood. Among them, communication heterogeneity is particularly critical because it increases communication cost and exacerbates the impact of the other two forms of heterogeneity. This thesis investigates neural network pruning as a central mechanism to mitigate communication costs in FL and develops a progressive research trajectory across increasingly complex environments. First, pruning is applied at the base system level to reduce communication cost while preserving model accuracy. In large-scale ITS scenarios, a dual pruning mechanism is proposed to adapt to intensified communication demands. To address statistical heterogeneity, a Gaussian Mixture Model (GMM)-based oversampling method is introduced to mitigate slightly skewed label distributions. Finally, a comprehensive pruning framework is developed for complex FL systems that jointly considers communication, statistical, and system heterogeneity. This work provides both practical solutions for reducing communication cost and a systematic perspective on the interaction of key challenges in FL, supporting scalable deployment in real-world systems.

Statement of Authorship Attribution

This thesis contains several chapters with material that was previously published or submitted during my Ph.D. candidature under the supervision of Dr. Wei Li. The ideas and the preparation behind each publication are primarily my own work, with all assistance that I received stated and acknowledged below:

Journal Articles

(1) Jiaming Pei, Wei Li; Shahid Mumtaz. *"From routine to reflection: pruning neural networks in communication-efficient federated learning,"* IEEE Transactions on Artificial Intelligence, 2024 . Pei *et al.* (2024) is reported in Chapter 4. I initiated the study, designed the algorithms, conducted the analysis and experiments, and drafted the manuscript.

(2) Jiaming Pei, Wei Li. *"Dual model pruning enables efficient federated learning in intelligent transportation systems,"* IEEE Transactions on Intelligent Transportation Systems, 2024. Pei and Li (2024a) is reported in Chapter 5. I initiated the study, designed the algorithms, conducted the analysis and experiments, and drafted the manuscript.

(3) Jiaming Pei, Wei Li. *"Unveiling the effects of slightly skewed labels on traffic data analysis,"* IEEE Transactions on Intelligent Transportation Systems, 2024. Pei and Li (2024b) is reported in Chapter 6. I initiated the study, designed the algorithms, conducted the analysis and experiments, and drafted the manuscript.

Conference Papers

(1) Jiaming Pei, Wei Li. *"Clustered Federated Learning with Slightly Skewed Labels,"* ICLR Tiny Paper, 2023 . Pei and Li (2023) is also reported in Chapter 6. I initiated the study, designed the algorithms, conducted the analysis and experiments, and drafted the manuscript.

Submitted Manuscripts

(1) Jiaming Pei, R. Venkatesha Prasad, Shahid Mumtaz and Wei Li. *"FedSP: Federated Segmented Pruning Learning for Enhanced Communication Efficiency in Industrial Cyber-Physical Systems,"* submitted to Nature Communications Engineering. This research is reported in Chapter 7. I initiated the study, designed the algorithms, conducted the analysis and part of the experiments, and drafted the manuscript.

Student Name: Jiaming Pei **Signature:**

Date: 17th March 2026

As the supervisor for the candidature upon which this thesis is based, I confirm that the authorship attribution statements above are correct.

Supervisor Name: Wei Li **Signature:**

Date: 17th March 2026

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Research Background

The rapid development of machine learning has been largely driven by advances in computing power and the widespread availability of massive datasets LeCun *et al.* (2015); Lehdonvirta *et al.* (2024). Traditionally, model training relies on a centralized paradigm, where data from diverse sources is collected in large-scale data centers or the cloud for optimization Kairouz *et al.* (2021). While this paradigm has enabled major breakthroughs in the early stages of deep learning, it faces structural bottlenecks that make it increasingly unsustainable in the era of massive and sensitive data. Centralizing raw data introduces severe privacy and compliance risks in domains such as healthcare, finance, and transportation, which are strictly regulated by laws including GDPR Voigt and Von dem Bussche (2017), HIPAA Act and others (1996), and CCPA Goldman (2020), while also amplifying users' concerns about privacy breaches Acquisti *et al.* (2015). Second, communication bottlenecks arise as data volumes grow explosively. Recent statistics indicate that the global datasphere reached approximately 181 zettabytes in 2025, much of which is generated at the network edge. For example, a self-driving car can generate terabytes of sensor data per day, making centralized uploading infeasible Shi *et al.* (2016). Third, latency constraints limit centralized learning in scenarios such as autonomous driving and industrial control, where millisecond-level responsiveness is critical Satyanarayanan (2017); Zhou *et al.* (2019). Finally, data silos persist across institutions and devices, where legal, competitive, and technical barriers prevent cross-domain data sharing, leading to limited and biased datasets Li *et al.* (2022a); Yang *et al.* (2019); Zhang *et al.* (2016). These structural challenges reveal that the traditional centralized

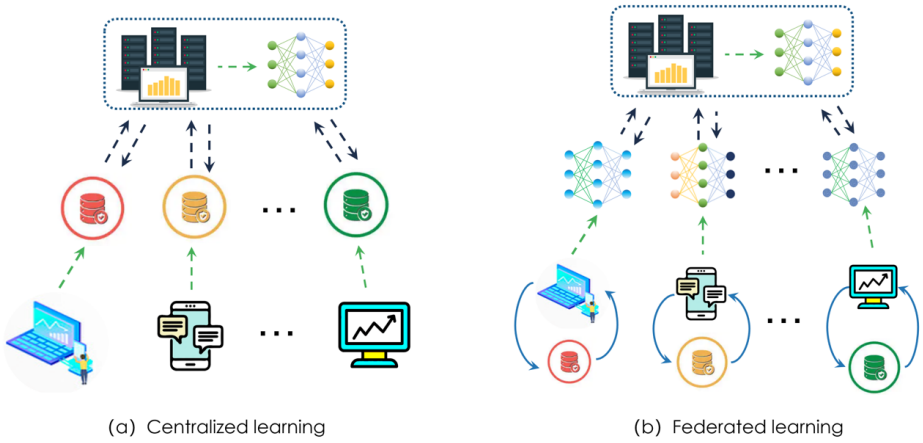


FIGURE 1.1. Comparison between centralized learning and FL. (a) In the centralized paradigm, raw data from multiple sources must be uploaded to the cloud for training, which raises concerns of privacy leakage, communication burden, and legal risks. (b) In FL, raw data remain on local clients and only model updates are communicated, enabling collaborative intelligence without exposing raw data.

paradigm is no longer sustainable in the era of massive and sensitive data. What is urgently needed is a new learning framework that can respect data privacy, reduce communication overhead, and enable collaboration across distributed data sources without requiring raw data centralization. To address these challenges, researchers have proposed a new paradigm known as **Federated Learning (FL)**.

1.2 Advantages and applications FL

FL is a distributed machine learning paradigm McMahan *et al.* (2017) that enables multiple clients—such as mobile devices, vehicles, hospitals, or organizations—to collaboratively train a shared global model under the coordination of a central server, while keeping their raw data localized, as illustrated in Figure 1.1(b). In contrast, the conventional centralized approach shown in Figure 1.1(a) requires data to be collected and stored in a data center. FL follows the principle of *"bringing the code to the data rather than the data to the code."*

Formally, each participating client performs local training on its private dataset and periodically transmits only model updates (e.g., gradients or parameter weights) to the server.

The server then aggregates these updates, typically through algorithms such as *Federated Averaging (FedAvg)*, to produce a new global model, which is redistributed back to the clients. This iterative process continues until convergence.

Through this design, FL achieves three essential goals simultaneously:

- **Privacy preservation:** Raw data never leaves the client, mitigating privacy risks and supporting compliance with data protection regulations.
- **Reduced communication overhead:** Transmitting compact model updates is significantly more efficient than uploading massive raw datasets.
- **Collaborative intelligence:** Organizations and devices that cannot or will not share data directly can still contribute to and benefit from a collective model.

Beyond these general advantages, FL has shown strong potential across a wide range of real-world applications. In **autonomous driving**, highly sensitive perception data generated by vehicles is difficult to centralize, yet FL enables multiple cars to collaboratively train more robust driving models without exposing raw sensor data. In **unmanned aerial systems (UAV)**, FL supports collaborative perception and intelligent decision-making among distributed drones, thereby enhancing swarm intelligence. In the **Internet of Things (IoT)**, vast amounts of heterogeneous data generated by edge devices can be efficiently exploited through FL, avoiding the bandwidth burden and privacy risks of transmitting raw data to the cloud. In **industrial cyber–physical systems (ICPS)**, FL helps break down industrial data silos and enables cross-device and cross-factory optimization, supporting predictive maintenance, process control, and system reliability. These applications highlight the practical value of FL as an enabler of distributed intelligence in privacy-sensitive and data-intensive domains.

1.3 Motivation and Challenges

Although federated learning has shown strong potential in domains such as autonomous driving, UAV networks, IoT, and ICPS, existing research mostly addresses a specific set of challenges. For example, some efforts focus on reducing communication overhead, others on

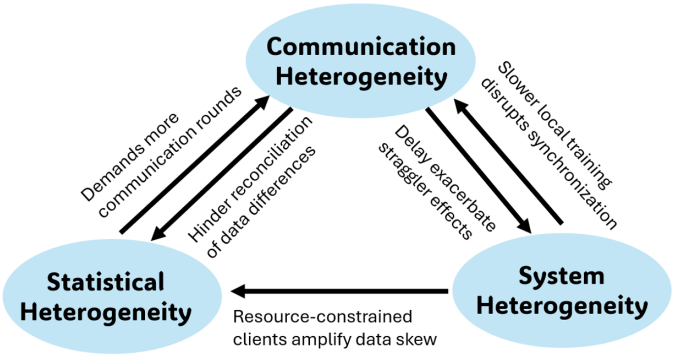


FIGURE 1.2. Interplay among communication, statistical, and system heterogeneity in FL systems.

mitigating the effects of non-IID data distributions (Non-IID) Bonawitz *et al.* (2019); Kairouz *et al.* (2021); Lu *et al.* (2024); Nishio and Yonetani (2019) , and still others on scheduling or fault tolerance under limited system resources. These lines of work, however, remain largely fragmented and lack a holistic understanding of how communication, statistical, and system heterogeneity are related and how they interact.

This fragmentation leads to two major limitations. First, optimizing along a single dimension often ignores coupling effects among challenges. For instance, reducing the frequency of communication rounds may temporarily relieve bandwidth pressure, but it simultaneously exacerbates convergence difficulties under statistical heterogeneity. Similarly, scheduling strategies that consider device heterogeneity but neglect communication conditions can still produce severe system bottlenecks. Second, without an integrated perspective, research outcomes are difficult to generalize to more complex application environments. In real-world systems, the three forms of heterogeneity almost always coexist and interact, which makes single-point solutions limited in both transferability and applicability.

The following subsections details each heterogeneity and examine how they influence one another as shown in Figure 1.2, establishing the foundation for the research trajectory developed in this thesis.

1.3.1 Communication heterogeneity

The first and most critical challenge is **communication heterogeneity**, which refers to the variability and instability of communication conditions across participating clients Bonawitz *et al.* (2019); Kairouz *et al.* (2021). In real-world deployments, clients are connected through highly diverse and often unreliable networks: mobile devices rely on 4G or 5G links with fluctuating bandwidth, vehicles communicate through vehicular-to-vehicular (V2V) or vehicular-to-infrastructure (V2I) channels that frequently experience disconnections, and IoT sensors often operate on low-power wireless networks with only a few kbps of capacity Gu *et al.* (2024); Nishio and Yonetani (2019). Such disparities mean that different clients may experience vastly different transmission speeds and latencies, ranging from hundreds of Mbps in industrial Ethernet to less than 5 Mbps in rural vehicular settings Hästbacka *et al.* (2021); Xu and Duan (2019). Federated learning further amplifies these issues because training requires **many rounds of iterative communication** between clients and the server McMahan *et al.* (2017). In each round, clients must upload their local model updates and download the updated global model. For modern deep neural networks, the size of these parameters is substantial. In many privacy-sensitive or distributed scenarios such as intelligent transportation systems, healthcare analytics, and industrial monitoring, raw data cannot be transmitted due to privacy regulations, data ownership constraints, or real-time requirements. These tasks often require large and expressive models to achieve satisfactory performance while introduces significant communication overhead. For example, ResNet-50 contains more than 25 million parameters, corresponding to roughly 100 MB in single precision. Transmitting such models over a 4G uplink (10–20 Mbps) requires 40–80 seconds per round, while in vehicular or IoT environments where uplink bandwidth may fall below 5 Mbps, the delay can exceed two minutes Sattler *et al.* (2019). Even under favorable Wi-Fi conditions (100 Mbps), each round still requires about eight seconds, which accumulates significantly over hundreds of rounds. These realities highlight why communication heterogeneity is not merely one challenge among others, but the **fundamental bottleneck** of federated learning. Unlike statistical or system heterogeneity, which influence the efficiency or fairness of optimization, communication is the essential mechanism by which learning progresses: without timely and

reliable exchange of model updates, global convergence cannot be achieved at all. As a result, communication heterogeneity directly determines the scalability of FL and often amplifies the negative impact of data and system heterogeneity.

1.3.2 Statistical heterogeneity

Statistical heterogeneity can be regarded as the intrinsic challenge of federated learning, since each client naturally generates and owns data with distinct characteristics. As a result, client datasets are rarely independent and identically distributed (non-IID) Kairouz *et al.* (2021); Zhang *et al.* (2024), and this internal diversity directly drives inconsistencies in how local updates contribute to global convergence. This heterogeneity manifests in several forms: (i) *label distribution skew*, where class proportions differ drastically across clients—for instance, vehicles in different regions encounter congestion and collision events with very different frequencies; (ii) *feature distribution skew*, where samples of the same class follow distinct feature distributions, such as camera viewpoints, illumination, or LiDAR density varying across driving environments Liu *et al.* (2021); (iii) *quantity skew*, where clients hold significantly different amounts of data, as in IoT settings where edge servers collect far more samples than low-power sensors; and (iv) *concept shift (or drift)*, where the mapping between features and labels differs across clients or evolves over time, as in industrial anomaly detection where fault patterns change with equipment aging Yang *et al.* (2019). The consequences of such non-IID distributions are twofold. First, they induce **client drift**, where local updates deviate from the global optimum, slowing convergence or even leading to divergence Karimireddy *et al.* (2020). Second, they degrade the generalization ability of the global model, as the aggregation is biased toward dominant clients or majority classes, which undermines fairness and reliability in safety-critical domains such as intelligent transportation systems. Importantly, statistical heterogeneity also interacts strongly with **communication efficiency**. Because convergence is inherently slower under non-IID conditions, more communication rounds are required to achieve a satisfactory model Li *et al.* (2020). In this way, statistical heterogeneity, as an internal factor, exacerbates the already severe communication bottleneck, further underscoring the need for approaches that jointly address both challenges.

1.3.3 System heterogeneity

System heterogeneity reflects disparities in the computational and memory resources available across participating clients. Devices involved in FL range from lightweight embedded controllers with only megaflop-level capacity to edge servers and cloud nodes capable of teraflop-level performance Xu and Duan (2019). Such differences mean that the time required for local training varies widely across clients, leading to inconsistent progress within each communication round. These disparities manifest in several ways. First, resource-constrained clients often become **stragglers**, delaying synchronization and slowing down global convergence. Second, mismatched training speeds can cause **synchronization failures** and instability in aggregation, as some clients lag behind while others finish early. Third, many industrial applications impose strict real-time requirements for inference and control tasks in the order of milliseconds. When federated learning training is executed on the same devices, it competes for computational resources with these real-time operations, creating a tension between iterative training processes and deterministic latency constraints Hästbacka *et al.* (2021); Peng *et al.* (2022). System heterogeneity also interacts with the other two challenges. Longer local training times worsen the impact of communication heterogeneity by making synchronization even more dependent on the slowest clients. Meanwhile, resource-limited devices may hold smaller or less diverse datasets. However, excluding such devices from training is often impractical in real-world federated learning systems because they may contain unique data collected in specific environments. Removing them could reduce data coverage and introduce additional bias. As a result, their participation can still amplify the effects of statistical heterogeneity.

1.3.4 Influence and relationship among three heterogeneity

To better illustrate the interplay among communication, statistical, and system heterogeneity, we conduct a simple experiment on MNIST using FedAvg. The goal is not to achieve state-of-the-art performance, but to highlight how different heterogeneity conditions affect convergence behavior and final model accuracy. Starting from a homogeneous and IID

baseline (Experiment 1), we progressively introduce communication heterogeneity (Experiment 2), then add statistical heterogeneity (Experiment 3), and finally combine all three by further introducing system heterogeneity (Experiment 4). In Experiment 1 (baseline), we simulated five clients on a single machine with homogeneous compute resources and IID data distribution. Communication was local, with no artificial constraints on bandwidth or latency. Experiment 2 extended this to two virtual machines, with three clients running on VM1 and two clients on VM2. The compute resources remained homogeneous, and the data distribution was IID. To introduce communication heterogeneity, VM1 was configured with 10 Mbps bandwidth and 10 ms latency, while VM2 was limited to 1 Mbps bandwidth and 50 ms latency. Experiment 3 maintained the same client distribution and compute setup as Experiment 2 but used a non-IID data partitioning strategy in which each client only held samples from two to three digit classes. In addition, we imposed harsher network conditions: VM1 remained at 10 Mbps and 10 ms, while VM2 was configured with 1 Mbps bandwidth, 100 ms latency. Finally, Experiment 4 introduced system heterogeneity on top of the settings of Experiment 3. Specifically, VM1 was equipped with a GPU and VM2 was CPU-only, while retaining the same non-IID partitioning and network constraints as in Experiment 3.

As summarized in Table 1.1, several observations can be made. When comparing Experiment 1 and Experiment 2, introducing communication heterogeneity while keeping IID data and homogeneous compute resources leads to a slight increase in training rounds and total communication volume, while the final accuracy remains largely unchanged (91% vs. 90%). This indicates that communication heterogeneity mainly affects training efficiency rather than model quality under IID conditions.

However, when statistical heterogeneity is introduced in Experiment 3, the impact becomes more pronounced. The number of communication rounds increases significantly (from 58 to 86), training time grows substantially, and the final accuracy drops to 86%. This suggests that non-IID data distributions make the optimization process more difficult, requiring additional communication rounds for convergence while also degrading model performance. Experiment 4 further introduces system heterogeneity on top of communication and statistical heterogeneity. In this case, the model accuracy decreases more dramatically to 78%, while both the

TABLE 1.1. FedAvg results on MNIST under different heterogeneity settings

	Configuration	Accuracy (%)	Rounds	Time (s)	Com. (MB)
1	IID, homogeneous compute, single machine	91	55	1094	110
2	IID, homogeneous compute, 2 VMs with heterogeneous network	90	58	1134	232
3	non-IID, homogeneous compute, 2 VMs with heterogeneous network + packet loss	86	86	1830	344
4	non-IID, heterogeneous compute (GPU + CPU), 2 VMs with heterogeneous network	78	99	2157	396

number of rounds and total training time continue to increase. The performance degradation reflects the combined effect of heterogeneous training speeds, unstable communication conditions, and non-IID data distributions. In particular, slower clients may delay synchronization and produce model updates of varying quality, which further amplifies the challenges already introduced by communication and statistical heterogeneity. these results illustrate that the three types of heterogeneity do not affect federated learning independently. Communication heterogeneity primarily influences training efficiency, statistical heterogeneity affects optimization stability and model quality, and system heterogeneity further exacerbates both effects. Their combined presence therefore creates a compounded impact on convergence behavior and final performance.

1.4 Research approach

In the previous experiments and discussions, we have demonstrated the close interplay between the three major heterogeneities of communication, data, and system. Communication heterogeneity manifests itself most directly in the form of communication overhead. In federated learning, when clients operate on heterogeneous networks, high-bandwidth nodes can rapidly complete transmissions, while low-bandwidth or high-latency nodes become bottlenecks, slowing down the overall system. As model size increases, this imbalance causes communication overhead to rise sharply, becoming the primary factor limiting the scalability

of federated learning. More importantly, unstable communication conditions not only reduce training efficiency but also amplify the convergence difficulties caused by data heterogeneity and the inconsistent synchronization caused by system heterogeneity. Therefore, this thesis takes the communication overhead issue as its starting point, starting from communication heterogeneity, and gradually reveals the relationships and interactions between the three major heterogeneities.

To alleviate the high communication overhead caused by heterogeneous communication settings, this thesis employs neural network pruning as its primary research tool. Pruning is a mature model compression technique that eliminates redundant parameters, thereby substantially reducing model size while preserving near-optimal accuracy. By lowering the number of bytes exchanged in each communication round, pruning alleviates communication bottlenecks at the source and improves convergence efficiency. In contrast to alternative compression or acceleration methods, pruning offers two distinctive advantages. First, it acts directly on the model without requiring specialized hardware, which makes it broadly applicable and portable. Second, pruning should not be viewed merely as an engineering mechanism for reducing communication cost. Instead, it provides a unifying research framework through which the interactions among communication heterogeneity, data heterogeneity, and system heterogeneity can be systematically investigated.

This thesis takes communication efficiency as its central thread and develops a progressive research trajectory from validating pruning effectiveness in a base system, to demonstrating its adaptability under more complex communication settings with statistical heterogeneity, and finally to systematizing the approach in environments where communication, statistical, and system heterogeneity coexist.

In the first stage, since global convergence relies on iterative parameter exchanges, per-round transmission cost can quickly dominate training efficiency. Chapter 4 proposes a server-side pruning algorithm based on parameter similarity, which reduces the size of transmitted updates while maintaining acceptable accuracy. Under controlled non-IID conditions, experiments verify that pruning achieves a favorable balance between communication efficiency and model performance. This stage establishes neural network pruning as the methodological baseline

for subsequent studies and lays the foundation for improving FL efficiency in more complex environments.

In the second stage, the number of clients is enormous, communication demands grow exponentially, and data distribution often exhibits skewed and imbalanced characteristics. Chapter 5 constructs a simulation environment with a large number of clients and introduces slight label skew and data imbalance to reflect statistical heterogeneity. In this setting, this paper proposes a dual pruning mechanism based on mutual information, which reduces communication traffic in both the upload and download directions while maintaining model accuracy. This design not only considers the differences in the number of samples between different clients but also introduces slightly skewed labels as a complementary form of data heterogeneity. We find that even small label distribution shifts can disrupt the training efficiency of federated learning. Chapter 6 designs a clustering and oversampling method based on the Gaussian mixture model (GMM) to effectively mitigate the negative impact of data imbalance on convergence. This phase not only demonstrates the coupled effect of communication and data heterogeneity but also verifies the applicability of pruning as a general strategy in complex scenarios, laying a solid foundation for research in the third stage.

The third stage focuses on more complex potential application scenarios such as ICPS. Compared with the previous stages, this setting incorporates all three forms of heterogeneity: diverse communication conditions, highly non-IID data distributions, and heterogeneous client resources in terms of computation and storage. Chapter 7 proposes a stage-aware unified pruning framework, Federated Segmented Pruning (FedSP), which dynamically adjusts pruning intensity according to the convergence state during training. By coupling pruning schedules with system dynamics, FedSP achieves a balance between communication cost and model accuracy, while maintaining convergence stability even under the coexistence of communication, statistical, and system heterogeneity.

1.5 Contributions

The main contributions of this thesis are summarized as follows:

- **System Perspective**

  One of the core contributions of this thesis is the introduction of a systematic research perspective that integrates communication, data, and system heterogeneity into a progressive framework for unified investigation. Through this perspective, the paper reveals the connections and interactions between the three major heterogeneities, clarifying how they jointly influence the convergence and performance of federated learning. This systematic framework not only addresses the fragmented nature of existing research but also provides theoretical guidance and methodological support for the scalable and robust deployment of federated learning in complex application scenarios.

- **Method Feasibility and Extensibility**

  We integrate neural network pruning into federated learning through parameter-similarity pruning, dual mutual-information pruning, and the stage-aware FedSP framework. These methods reduce communication cost while preserving accuracy, demonstrating both feasibility and adaptability under diverse heterogeneities.

- **Theoretical Guarantees**

  We establish rigorous convergence analysis showing that pruning-based federated optimization reaches ε-stationary points under standard assumptions. The results quantify the trade-offs between sparsity, data heterogeneity, and accuracy, and ensure stable convergence when applying dynamic pruning strategies.

- **Extension to Data Heterogeneity**

  This work extends beyond imbalance by slightly skewed labels. Through GMM-based clustering and oversampling, we mitigate negative effects and deepen understanding of federated learning behavior under more complex distributions.

1.6 Organization

The remainder of this thesis is structured as a progressive journey from theoretical foundations to increasingly complex heterogeneous environments. Chapter 2 introduces the necessary preliminaries, including the core principles of federated learning, the concept of non-IID

data, and neural network pruning techniques. Chapter 4 focuses on base FL systems, demonstrating that communication overhead remains the dominant bottleneck even under idealized conditions, and presents a pruning-based framework to mitigate this issue. Building on this foundation, Chapters 5 and 6 extend the investigation to scenarios characterized by both communication and statistical heterogeneity: Chapter 5 addresses communication efficiency through a dual pruning strategy, while Chapter 6 tackles label skew and class imbalance with a Gaussian mixture model (GMM)-based oversampling method. Chapter 7 further advances the study to environments where communication, data, and system heterogeneity coexist, and introduces Federated Segmented Pruning (FedSP), a stage-aware framework that dynamically adjusts pruning intensity to maintain stable convergence. Finally, Chapter 8 summarizes the main contributions, reflects on their broader implications, and outlines promising directions for future research in heterogeneous FL systems.

CHAPTER 2

Technical Background and Foundations

Before delving into our proposed methodology, it is necessary to revisit several fundamental concepts and techniques that provide the theoretical foundation for subsequent system design and algorithmic analysis. The challenges addressed in this work primarily arise from the deployment of FL in heterogeneous real-world environments: on the one hand, the high complexity of neural networks imposes heavy communication and storage burdens, which motivates the use of pruning and other model compression techniques to improve system efficiency; on the other hand, distributed data are often non-independent and identically distributed (non-IID), leading to difficulties in both convergence and performance stability. In addition, federated optimization must be grounded in classical machine learning training and optimization theory to ensure stable convergence under resource and communication constraints. Therefore, this chapter systematically reviews the relevant foundations, including the basic framework and challenges of FL, neural network pruning and model compression methods, the definition and common strategies for handling non-IID data, as well as the core principles of optimization and training. These discussions establish the necessary theoretical and technical background for the methods and experiments presented in the following chapters.

2.1 FL Fundamentals

FL is a decentralized machine learning paradigm that enables multiple clients to collaboratively train a shared model without exchanging their raw data. In its standard formulation, the global learning objective can be expressed as a distributed empirical risk minimization problem:

$$\min_{w \in \mathbb{R}^d} F(w) = \sum_{i=1}^{n} p_i F_i(w), \tag{2.1}$$

where n is the total number of clients, $p_i = \frac{n_i}{\sum_{j=1}^{n} n_j}$ denotes the relative weight of client i based on the size of its local dataset n_i, and

$$F_i(w) = \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(w; x_j^i, y_j^i) \tag{2.2}$$

represents the empirical loss of client i on its local dataset $\mathcal{D}_i = \{(x_j^i, y_j^i)\}_{j=1}^{n_i}$, with $\ell(\cdot)$ being the loss function.

A representative baseline algorithm in FL is **Federated Averaging (FedAvg)** Sun *et al.* (2022), as shown in Algorithm 1. In FedAvg, each selected client performs E steps of local stochastic gradient descent (SGD) based on the current global model w_t:

$$w_{t+1}^i \leftarrow w_t - \eta \sum_{e=1}^{E} \nabla F_i(w_{t,e}^i), \tag{2.3}$$

where η is the local learning rate and $w_{t,e}^i$ denotes the intermediate model at local epoch e on client i. The server then aggregates the updates from all participating clients:

$$w_{t+1} \leftarrow \sum_{i \in S_t} p_i w_{t+1}^i, \tag{2.4}$$

where $S_t \subseteq \{1, \ldots, n\}$ is the subset of clients selected at communication round t.

Despite its simplicity and effectiveness, FedAvg and related algorithms encounter significant challenges in real-world FL systems. These include *statistical heterogeneity* (i.e., non-IID data distributions across clients), *system heterogeneity* (variation in computing and communication capabilities), and limited or unreliable communication. To address these issues, various extensions have been proposed, such as FedProx, which introduces a proximal term to improve stability, and FedNova, which normalizes updates to reduce objective inconsistency.

Algorithm 1 Federated Averaging (FedAvg)

Input: Number of clients n, local epochs E, learning rate η, total communication rounds T.
Output: Final global model w_T
Server executes:
Initialize global model w_0
for each round $t = 0, 1, 2, \ldots, T-1$ **do**
  Select a subset of clients $S_t \subseteq \{1, \ldots, n\}$
  for each client $i \in S_t$ **in parallel do**
    $w_i^{(t+1)} \leftarrow$ **ClientUpdate**(w_t, η, E)
  end for
  Aggregate client updates:

$$w_{t+1} \leftarrow \sum_{i \in S_t} \frac{n_i}{\sum_{j \in S_t} n_j} w_i^{(t+1)}$$

end for
ClientUpdate(w, η, E)
$w^{(0)} \leftarrow w$
for local epoch $e = 1, \ldots, E$ **do**
  $w^{(e)} \leftarrow w^{(e-1)} - \eta \cdot \nabla F_i(w^{(e-1)})$
end for
return $w^{(E)}$

2.2 Neural network pruning

Neural network pruning is a technique that aims to reduce the complexity of deep learning models by removing unnecessary or redundant components such as weights, neurons, or channels Blalock *et al.* (2020); Sietsma and Dow (1988). The goal is to make the network more efficient in terms of memory usage and inference speed, while maintaining or minimally degrading its original performance Molchanov *et al.* (2019). Pruning is particularly valuable for deploying models on resource-constrained devices like mobile phones or embedded systems.

The core idea behind pruning is that over-parameterized neural networks often contain redundant structures Shi *et al.* (2024). After training, many parameters have negligible impact on the final output. By identifying and removing these parameters, we can simplify the network without significantly affecting accuracy.

The general principle involves:

- **Redundancy assumption**: Many weights or neurons are not critical for model predictions Wang *et al.* (2021b).
- **Importance evaluation**: Various criteria such as weight magnitude, activation statistics, or gradient information are used to estimate importance Lian *et al.* (2024).
- **Pruning and fine-tuning**: The network is pruned based on importance scores, then fine-tuned to recover any lost performance Xu *et al.* (2021a).

Pruning methods can be categorized based on the **granularity** of the pruned units and the **timing** of pruning application, as summarized in Table 2.1.

TABLE 2.1. Classification of neural network pruning methods by granularity and timing.

Category	Definition	Example
Unstructured Pruning Wang and Zhang (2024)	Removes individual weights from the network without changing its structure; produces sparse matrices.	Magnitude pruning: removes weights with the smallest absolute values.
Structured Pruning He and Xiao (2023)	Removes entire neurons, channels, filters, or layers; changes the network architecture.	Channel pruning: deletes less important output channels from convolutional layers.
Post-training Pruning Luo *et al.* (2017)	Applied after full model training; typically followed by fine-tuning to recover performance.	Prune a trained ResNet model, then fine-tune it on the original dataset.
During-training Pruning Shi *et al.* (2024)	Pruning is integrated into the training loop; weights are dynamically pruned during training.	Dynamic pruning with regularization techniques to enforce sparsity during training.

The general workflow for pruning a neural network includes the following steps:

(1) **Train the original model**: Start with a fully trained, high-performance baseline model.
(2) **Evaluate importance**: Use pruning criteria (e.g., weight magnitude, L1/L2 norm) to assess the significance of different parameters or structures.
(3) **Apply pruning**: Remove parameters or units based on a threshold or pruning ratio.

(4) **Fine-tune the pruned model**: Retrain the model on the original dataset to recover or maintain its accuracy.

(5) **Iterative pruning (optional)**: Repeat the prune–fine-tune cycle to achieve higher compression ratios.

Neural network pruning is a powerful model compression method that reduces the size and computational cost of deep networks. It leverages the redundancy in over-parameterized models by removing insignificant weights or structures. Pruning methods can be either unstructured or structured, and applied either post-training or during training. A typical pruning pipeline involves training, importance evaluation, pruning, fine-tuning, and optional iterative refinement. Properly applied, pruning can enable deep learning models to run efficiently on low-resource environments without sacrificing much performance.

2.3 Non-IID

In traditional centralized machine learning, it is typically assumed that training data is Independent and Identically Distributed (IID) Arafeh *et al.* (2022). However, in FL settings, data on each client often comes from diverse sources and exhibits personalized behaviors or preferences, leading to **Non-Independent and Identically Distributed (Non-IID)** data Torra (2023). This means the local data distributions vary significantly across clients. For instance, one user's phone might primarily capture pet photos, while another user mainly stores food images. These distributional differences create heterogeneity in data, which directly affects the training dynamics.

Non-IID data introduces several critical issues in federated optimization and global model performance:

- **Slower or unstable convergence:** Due to conflicting gradient updates from heterogeneous clients, the global model may converge slowly or even diverge.
- **Poor generalization:** The global model might perform poorly on unseen or underrepresented distributions, limiting its utility.

- **Client unfairness:** Clients with minority data distributions may see degraded performance, raising concerns of fairness and inclusiveness.
- **Reduced robustness:** Extreme data disparities can lead to overfitting to dominant distributions or poor adaptation to distribution shifts.

The Non-IID nature in FL can be categorized into several distinct types, as summarized in the Table 2.2 as below:

TABLE 2.2. Common Types of Non-IID in Federated Learning

Type	**Description**	**Example**
Label Distribution Skew	Clients have different sets or proportions of labels	One client only has images of cats, another only has dogs
Feature Distribution Skew	The distribution of input features differs across clients	Handwriting styles differ for digit recognition
Quantity Skew	Clients have vastly different amounts of data	One user has 10,000 samples, another has only 50
Concept Shift	The same input corresponds to different labels on different clients	A review is labeled "positive" by one client and "negative" by another

Non-IID data is an inherent and challenging characteristic of real-world federated learning systems. It can negatively impact model convergence, generalization, fairness, and robustness. Understanding the various types of Non-IID distributions is critical for designing effective FL algorithms. Addressing these challenges is essential for deploying federated systems at scale and in production environments.

2.4 Formal Problem Definition

We now formalize the optimization problem of federated learning (FL) under heterogeneous conditions. The global empirical risk minimization objective in FL can be extended as follows:

$$\begin{aligned} \min_{w \in \mathbb{R}^d} \quad & F(w) + \lambda_c \mathcal{C}(w) \\ \text{s.t.} \quad & \mathcal{D}(w) \leq \epsilon_d, \\ & \mathcal{S}(w) \leq \epsilon_s, \end{aligned} \tag{2.5}$$

where $F(w)$ denotes the global empirical risk and λ_c balances accuracy against communication cost. The additional terms $\mathcal{C}(w)$, $\mathcal{D}(w)$, and $\mathcal{S}(w)$ correspond to three major sources of heterogeneity in real-world FL systems. Specifically, $\mathcal{C}(w)$ captures communication heterogeneity, where clients differ in bandwidth and latency. If client k has bandwidth B_k, the incurred cost is $\mathcal{C}_k(w) = \frac{|w|}{B_k}$ and the total overhead is $\mathcal{C}(w) = \sum_{k=1}^{K} \mathcal{C}_k(w)$. $\mathcal{D}(w)$ measures statistical heterogeneity due to non-IID data distributions across clients, constrained by $\mathcal{D}(w) \leq \epsilon_d$ with ϵ_d representing the tolerated divergence. Finally, $\mathcal{S}(w)$ represents system heterogeneity, reflecting differences in computation, memory, and availability, with the constraint $\mathcal{S}(w) \leq \epsilon_s$ where ϵ_s is the tolerance threshold.

Communication heterogeneity is incorporated into the objective function because it directly represents a cost that must be minimized jointly with the learning objective. In contrast, statistical and system heterogeneity reflect properties of data distributions and device capabilities that cannot be directly optimized but must be tolerated within acceptable limits. Therefore, they are modeled as constraints. In practice, the parameters λ_c, ϵ_d, and ϵ_s are application-dependent. The coefficient λ_c controls the trade-off between model accuracy and communication cost and is typically determined through empirical tuning. The thresholds ϵ_d and ϵ_s represent acceptable levels of statistical divergence and system variability, respectively, and are set according to system requirements and resource conditions.

This unified formulation provides the mathematical foundation for the remainder of this dissertation. Chapter 4 first addresses communication overhead $\mathcal{C}(w)$ in idealized settings. Chapters 5–6 extend the study to more complex environments with explicit statistical constraints $\mathcal{D}(w) \leq \epsilon_d$. Finally, Chapter 7 incorporates compounded heterogeneity by introducing the system constraint $\mathcal{S}(w) \leq \epsilon_s$.

CHAPTER 3

Related Work

3.1 FL under Heterogeneity

FL has become a promising framework for distributed model training without sharing local data, yet real-world deployment inevitably encounters multiple forms of heterogeneity. Existing studies typically focus on a single dimension—communication, statistical, or system heterogeneity—while few attempt to jointly model and optimize across all three. Communication heterogeneity arises from inconsistent network conditions such as bandwidth, latency, and packet loss across clients. These disparities lead to asynchronous updates, communication delays, and global model instability. Solutions including gradient compression, quantization, and hierarchical aggregation Alistarh *et al.* (2017); Bernstein *et al.* (2018); Reisizadeh *et al.* (2020) reduce bandwidth demand, but they largely ignore the interplay between data distribution and computation constraints. Statistical heterogeneity, or non-IID data, originates from discrepancies in local data distributions. This challenge causes convergence slowdown and biased aggregation. Algorithms such as FedProx Li *et al.* (2020), FedPer Arivazhagan *et al.* (2019), and FedBoost Hamer *et al.* (2020) mitigate this problem by introducing proximal regularization, personalized local adaptation, or weighted aggregation. However, these methods assume homogeneous system and communication environments, which is rarely true in practice. System heterogeneity reflects variations in device computational power, memory, and energy capacity. Clients with limited resources may perform fewer updates or train smaller subnetworks, resulting in unbalanced global aggregation. System-aware frameworks such as Oort Lai *et al.* (2021), FedCS Nishio and Yonetani (2019), and DepthFL Kim *et al.* (2022) improve efficiency through adaptive scheduling and resource allocation but treat heterogeneity

as an isolated system-level issue. Most FL studies address one dimension of heterogeneity at a time, leaving a research gap in unified modeling and cross-dimensional optimization. This motivates our work to design a framework that concurrently tackles communication, statistical, and system heterogeneity through integrated pruning and aggregation mechanisms.

3.2 Model Compression and Neural Network Pruning

Model compression techniques, especially neural network pruning, have been widely used to reduce computational and communication overhead while preserving model accuracy. Traditional pruning strategies can be divided into unstructured and structured approaches. Unstructured pruning removes individual weights with small magnitudes Han *et al.* (2015); Hassibi and Stork (1992), producing sparse models but often resulting in irregular memory access and limited hardware acceleration. Structured pruning, by contrast, removes entire filters, channels, or layers Luo and Wu (2017); Meng *et al.* (2020), yielding models that are both lightweight and hardware-friendly. The Lottery Ticket Hypothesis (LTH) Frankle and Carbin (2019) demonstrated that subnetworks within large models can independently achieve competitive performance, inspiring initialization-aware methods such as SNIP Lee *et al.* (2018), GraSP Wang *et al.* (2019), and SynFlow Tanaka *et al.* (2020). Neural network pruning has been increasingly integrated into FL to enhance communication and training efficiency. SNIP Lee *et al.* (2018) pre-determines the importance of connections before training, reducing the need for iterative pruning. PrunFL Jiang *et al.* (2022a) performs client-side pruning early and gradually incorporates global updates to minimize computation. FedDUAP Zhang *et al.* (2022a) introduces adaptive server-side pruning using shared data, while DepthFL Kim *et al.* (2022) prunes deeper network layers based on device capacity. EFLMP Wu *et al.* (2023b) performs server-side reconstruction and parameter reuse, and Sub-FedAvg Vahidian *et al.* (2021) assigns shared subnetworks to clients with similar label distributions.

While these approaches still exhibit several limitations in federated learning environments. First, some methods rely on the availability of shared or auxiliary data at the server to guide

pruning decisions. In practice, such data may not be available due to privacy restrictions or deployment constraints, limiting the applicability of these approaches in real-world FL systems. Second, several methods introduce device-specific model structures or adaptive subnetworks tailored to heterogeneous client capabilities. While this can improve local efficiency, it often complicates model aggregation at the server and may lead to structural inconsistencies across clients. Third, certain approaches require additional coordination among clients or frequent model restructuring during training, which increases system complexity and may introduce instability during the aggregation process.

These limitations reveal that designing an effective pruning strategy for federated learning is non-trivial. The pruning mechanism must simultaneously reduce communication overhead, maintain model consistency across heterogeneous clients, and ensure stable convergence during iterative training. Balancing these objectives is particularly challenging in practical FL systems where communication resources, data distributions, and device capabilities vary significantly across clients. Therefore, there remains a need for pruning strategies that can achieve communication efficiency while preserving training stability and scalability in heterogeneous federated environments.

3.3 Communication-efficient FL

Communication overhead has long been recognized as one of the most critical bottlenecks in FL, especially when large-scale or resource-constrained edge devices participate in collaborative training. As each communication round involves transmitting model parameters or gradients between the clients and the central server, reducing transmission volume and frequency is essential for enabling scalable FL deployment.

To address this challenge, a wide range of communication-efficient optimization techniques have been proposed, including gradient compression, quantization, sparsification, and periodic aggregation. Among the earliest methods, SignSGD Bernstein *et al.* (2018) and QSGD Alistarh *et al.* (2017) demonstrated that sending only the signs or low-bit quantized representations of gradients can significantly reduce communication costs without substantial accuracy

loss. Subsequent works extended these principles by introducing adaptive quantization, where the quantization level changes dynamically according to training progress or gradient variance—such as AdaQuantFL Jhunjhunwala *et al.* (2021) and AQG Mao *et al.* (2022). Similarly, Deep Gradient Compression (DGC) Lin *et al.* (2018) and sparse communication techniques Aji and Heafield (2017) reduce communication frequency by transmitting only the most important gradient elements while applying momentum correction and gradient accumulation to preserve convergence stability.

Beyond gradient-level compression, periodic and local aggregation methods have also been widely adopted to further reduce the number of communication rounds. For instance, FedPAQ Reisizadeh *et al.* (2020) combines periodic model averaging with quantized updates, balancing local computation and communication efficiency. FedDM Xiong *et al.* (2023) refines this concept by introducing iterative distribution matching to align the global and local update distributions, thus reducing both transmission volume and bias accumulation. Recent studies such as FedCSTQ Zheng and Tang (2025) and Oh et al. (2021) integrate compressed sensing and quantization to further enhance compression ratios while maintaining model fidelity. Meanwhile, hybrid approaches like FedSparse Li *et al.* (2024) and SpaFL Kim *et al.* (2024) introduce sparsity constraints directly into the optimization objective, enabling simultaneous communication and computation efficiency.

3.4 Data Heterogeneity and Non-IID Mitigation

Data heterogeneity—one of the most critical challenges in FL—typically manifests as label skew, quantity skew, and feature shift Zhao *et al.* (2018). Label skew occurs when different clients possess data from distinct classes, quantity skew refers to imbalanced data volumes, and feature shift denotes distributional differences in feature space caused by contextual or environmental factors. To mitigate these issues, several strategies have been proposed: (1) Data balancing and resampling, including oversampling underrepresented labels or applying data augmentation and clustering-based reweighting Li *et al.* (2021a). (2) Personalized models,

such as FedPer Arivazhagan *et al.* (2019) and FedBN Li *et al.* (2021b), which maintain client-specific parameters while sharing common components. (3) Aggregation weighting, as seen in FedAvg McMahan *et al.* (2017), FedProx Li *et al.* (2020), and FedBoost Hamer *et al.* (2020), where local losses or data sizes are used to adjust global model updates. Beyond these direct approaches, Clustered Federated Learning (CFL) has recently emerged as an effective paradigm for mitigating data heterogeneity. CFL combines clustering and federated optimization to simultaneously enhance privacy protection, data balance, and model accuracy in decentralized environments. For instance, FedCluster Chen *et al.* (2020) and ICFA Ghosh *et al.* (2020) employ clustering algorithms to divide participating devices into multiple clusters, performing localized model updates within each cluster before aggregation, thereby enhancing model personalization and convergence speed. In contrast, FedKM Kumar *et al.* (2020) integrates encryption techniques to ensure secure data sharing and incorporates both local and global clustering phases to strengthen privacy protection and learning stability. FSC Thakkar and Joshi (2023) promotes efficient information exchange among devices using spectral clustering, improving the precision and robustness of cluster formation. Moreover, FCOT Farnia *et al.* (2022) introduces optimal transportation schemes to optimize communication and model update efficiency across clusters. CFC Sattler *et al.* (2020), a cooperation-based CFL algorithm, dynamically partitions devices into clusters and performs FL within each cluster, leveraging selected devices for global aggregation to maintain model consistency. FSSC Wang *et al.* (2021a) adopts a semi-supervised learning paradigm, utilizing labeled data to guide model updates that are then propagated to other devices, improving performance under label-scarce conditions. Finally, FedMAC Nguyen *et al.* (2024) aggregates models from multiple clusters through weighted averaging to generate an adaptive and more generalized global model.

3.5 System-level Optimization in FL

At the system level, heterogeneity in hardware capabilities, network latency, energy budgets, and participation frequency further complicates the deployment and scalability of FL. Hardware and latency heterogeneity often lead to the so-called straggler effect, where slower or

resource-constrained clients delay global aggregation and hinder convergence. To address these challenges, several adaptive scheduling and resource-aware frameworks have been proposed. Oort Lai *et al.* (2021) prioritizes clients that contribute the most to model accuracy per unit of communication cost, balancing statistical utility and system efficiency. FedCS Nishio and Yonetani (2019) dynamically selects participating clients based on real-time availability and estimated completion time, reducing idle waiting caused by slow devices. FedMAC Nguyen *et al.* (2024) extends this idea by employing cluster-based aggregation and adaptive communication frequency to enhance fairness and model generalization across heterogeneous devices. Energy efficiency and dynamic participation have also received growing attention, as edge devices often operate under tight energy and computation constraints. Energy-aware methods, such as those proposed by Wang et al. Wang *et al.* (2021a), optimize client selection and local update frequency to minimize total energy consumption without compromising convergence. Hierarchical or multi-tier FL architectures further improve scalability by introducing intermediate aggregation layers (e.g., edge or fog nodes) that reduce overall communication load and system latency. In industrial and intelligent transportation systems, integrating FL with edge computing and blockchain technologies has been shown to enhance reliability, traceability, and data privacy Aflaki *et al.* (2024); Guendouzi *et al.* (2022). These system-level approaches collectively contribute to improved scalability and robustness of FL in practical, resource-diverse environments. However, most existing works treat hardware and system heterogeneity in isolation and rarely consider their interdependence with statistical and communication heterogeneity, underscoring the need for a unified optimization framework that jointly models data, communication, and system dimensions—a gap that this dissertation aims to address.

CHAPTER 4

Proof of Concept of Neural Network Pruning: Communication-Efficient Base System

---

In the previous chapter, we presented a unified formalization of FL under heterogeneous conditions, where the global optimization problem incorporates three major factors: communication overhead $\mathcal{C}(w)$, statistical heterogeneity $\mathcal{D}(w)$, and system heterogeneity $\mathcal{S}(w)$. Within this framework, communication overhead emerges as the most fundamental bottleneck. This chapter focuses on addressing $\mathcal{C}(w)$ on base system. We propose a server-side pruning algorithm based on parameter similarity. The method leverages the differences between the global model and local models to guide the pruning process, thereby reducing the size of transmitted updates. In addition, we examine how different pruning schedules affect model performance. Experiment results demonstrate that the proposed method preserves model accuracy under non-IID data conditions while reducing communication cost by 50%–70%. Moreover, by performing pruning on the server, part of the computational burden is shifted away from clients, alleviating device-side constraints and lowering overall energy consumption. In summary, this chapter targets the communication overhead term $\mathcal{C}(w)$ in the unified formulation, validating neural network pruning as an effective strategy to improve communication efficiency in a prototype FL system. This provides the methodological foundation for subsequent chapters, which address more complex heterogeneous conditions.

4.1 Introduction

Ensuring communication efficiency in FL heavily depends on the size of the neural network model. Smaller models can be transmitted between devices more easily, minimizing

communication overhead. While over-parameterization in neural networks improves model generalization, it also demands significant resources Poyatos *et al.* (2023), an issue that persists in FL Chen *et al.* (2023); Yu *et al.* (2023), where updating local parameters involves substantial communication and computation. Neural network pruning Shang *et al.* (2022) offers an effective solution by reducing model size without compromising functionality.

The core idea of network pruning is to assess the importance of neural network model parameters based on their training effect on a single dataset Yu *et al.* (2022). In FL, datasets are often unevenly distributed across clients, making a single client's data an unreliable reflection of the global distribution. This uneven distribution hampers the generalization ability of neural network models, leading to overfitting on small samples and underfitting on the global task Wan *et al.* (2022). The optimal sub-network of the global model can be identified early in the updating process Rachwan *et al.* (2023). During pruning, some parameters may be removed to enhance local model performance, even though they might be vital for the global model, particularly in non-IID scenarios. The weighted average aggregation mechanism can cause these performance-oriented parameters to slow down convergence. Consequently, after aggregation, the global model may fail to achieve optimal performance due to the absence of pruned parameters. Pruning on clients can improve local performance but may not necessarily enhance global results Kim *et al.* (2022); Wu *et al.* (2023b). Moreover, most pruning-related studies focus only on pruning itself, neglecting device heterogeneity. Performing pruning on the client side leads to identical processes across clients, creating computational redundancy. Given the limited computational resources of client devices, this approach can be highly resource-intensive Jiang *et al.* (2023); Salehi *et al.* (2024); Yi *et al.* (2024).

Theoretically, since the server has global knowledge, each local model can remove parameters that contribute less to the global model with the help of the latest aggregated global model. This reduces the difference in intrinsic parameters between the global and local models while still maintaining model performance. Servers are typically more resource-rich than clients, and by performing pruning operations on the server, client resources can be conserved. This approach achieves better load distribution by assigning computation-intensive pruning tasks

to appropriate hardware, thereby optimizing overall resource utilization Xiong *et al.* (2023); Zhang *et al.* (2022b).

In practice, many server-side pruning methods Kim *et al.* (2022); Vallapuram *et al.* (2022); Wu *et al.* (2023b) do not always outperform client-side pruning, especially in IID or non-IID scenarios. A key reason for this performance discrepancy is the absence of a step to repair the global model after pruning on the server side. During pruning, some globally important information may inadvertently be removed, necessitating a subsequent repair step to restore model performance.

In this chapter, our approach focuses on pruning neural network models at the server while leveraging global information. To address the challenges of server-side pruning, we modify the order of model pruning to **Aggregation-Pruning-Aggregation**. With this new order, local models benefit from the latest global model, which is aggregated from them in each round. This iterative pruning process ensures that only the least important parameters are removed, enhancing the overall efficiency and effectiveness of the pruning. In contrast, conventional client-side pruning methods prune neural network models based on the accuracy of individual local models on a single client. This strategy may not align with the broader interests of the entire FL system, as what significantly impacts one client may not be universally beneficial. Our method takes a different approach by treating the global model as a threshold matrix. We prune local models by evaluating the similarity between the local and global models, specifically trimming parameters in local models that show significant differences from the global model. This strategy helps the global model converge more quickly without compromising accuracy, thereby improving the overall efficiency of the FL system.

- **Where to prune.** We introduce a new idea for neural network pruning in FL, where the local models are centrally pruned based on the similarity between the global and local models. Our pruning method can also speed up the convergence in FL.
- **How to prune.** Following the 'Aggregation-Pruning-Aggregation' order, our method prunes local models compared to the latest global model and re-aggregates the pruned local models to an updated global one. This strategy exploits global knowledge to remove redundant parameters from neural networks efficiently.

- **Gain from the pruning.** Our results indicate that in FL systems, server-side pruning achieves faster convergence, especially when the number of communication rounds is fewer than the number of participating clients. The experiment results also show that our method outperforms the benchmark algorithms on well-known data sets in multiple ways. We also theoretically proved that our pruning algorithm outperforms state-of-the-art algorithms in both convergence rate and communication complexity.
- **Optimizing Resources Usage via Pruning.** We showcase that executing our pruning method on the CPU allows the GPU to concentrate on training, leading to efficient system resource allocation. This mirrors the resource disparity in real-world FL between servers and client devices, enhancing the overall system efficiency.

4.2 Methodology

4.2.1 Problem Statement

In FL, the server S and N clients jointly train a global model w^g. Specifically, the server aggregates the local model w^c after e epochs of local training, and the training lasts for T rounds of communication. The objective function can thus be written as follows:

$$\min_{w^g \in \mathbb{R}^d} F(w)F(w^g) = \sum_{n=1}^{N} \frac{D^n}{D} F_n(w^c), \tag{4.1}$$

where D^n denotes the number of samples of client n and F_n represents the loss function of client n's training.

To determine the local model w^c, the stochastic gradient descent (SGD) algorithm is used to train the local model with the learning rate η. It is defined as follows:

$$w^c(t+1) \leftarrow w^c(t) - \eta \cdot \nabla F_n(w^c(t)). \tag{4.2}$$

The pruning target is the weight w^c. Let the pruning function be $G(\alpha, w^c) \sim \Lambda_t$, where α is the pruning rate and Λ_t represents the threshold in the t-th round of updates. $m(n)$ denotes

all non-zero values of the mask. The weight w^c after pruning can be denoted as:

$$w^{c'} \leftarrow w^c \oplus m(n). \tag{4.3}$$

Thus, the objective function is transformed into:

$$\min_{w^g \in \mathbb{R}^d} F(w^g) = \sum_{n=1}^{N} \frac{D^n}{D} F_n(w^{c'}). \tag{4.4}$$

4.2.2 The Pruning Method

Unlike many other methods that prune models on the clients, our pruning procedure takes place on the central server during model aggregation. Therefore, the global model update in our method can be represented as:

$$w^g = agg(G(\alpha, w_1^c), G(\alpha, w_2^c), ..., G(\alpha, w_N^c)). \tag{4.5}$$

The main task is to search for the pruning function G, and we aim to prune all layers in the network. Our search process includes two steps: model preprocessing, as described by Equations (4.6) to (4.9), and pruning operations, as described by Equations (4.10) to (4.15). The pseudocode for the pruning process is shown in Algorithm 5.

Batch normalization is widely used in neural networks. During training, the mean u_B and standard deviation σ_B of the input data for each mini-batch B can be computed as follows:

$$u_B = \frac{1}{s_B} \sum_{i=1}^{s_B} x^{(i)}, \ \sigma_B = \sqrt{\frac{1}{s_B} \sum_{i=1}^{s_B} (x^{(i)} - u_B)^2}, \tag{4.6}$$

where $x^{(i)}$ denotes the i-th sample in the mini-batch and s_B is the batch size. Through zero-mean normalization, $x^{(i)}$ can be represented as:

$$\tilde{x}^{(i)} = \frac{x^{(i)} - u_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \tag{4.7}$$

[1]The value of ε is often set to 10^{-4} to avoid a zero denominator.

Algorithm 2 Neural network pruning on the server

Input: Pruning rate α, Number of clients N, Updating rounds T, learning rate η, and epoch E.
Output: Global model w^g.
Server:
Initialize global model w_0^g, Difference Matrix M
for each round $t = 1, 2, 3, ..., T$ **do**
  $M \leftarrow 0$
  for each client $n = 1, 2, 3, ..., N$ **do**
    $w_n^c(t+1) \leftarrow$ **ClientUpdate** $(w_n^c(t), w^{ave}(t))$
  end for
  $w^{ave}(t+1) \leftarrow \sum_{n=1}^{N} \frac{D^n}{D} w_n^c(t+1)$
  $M \leftarrow \sum_{n=1}^{N} |w_n^c(t+1) - w^{ave}(t+1)|$
  Sort M in descending order
  for each client $n = 1, 2, 3, ..., N$ **do**
    if $M[i][j]$ in the largest α of parameters **then**
      $w_{t+1,n}^c[i][j] \leftarrow 0$
    end if
  end for
  $w^g(t+1) \leftarrow \sum_{n=1}^{N} \frac{n^n}{n} w_n^c(t+1)$ Only non-zero parameters.
end for
ClientUpdate $(w_n^c(t), w^{ave}(t))$
for each local epoch $e = 1, 2, ..E$ **do**
  $w_n^c(t+1) \leftarrow w_n^c(t) - \eta \cdot \nabla F_n(w_n^c(t)) - \frac{\eta\lambda}{n} w_n^c(t)$
  if $(w_n^c(t+1)) \geq 0$ **then**
    $w_n^c(t+1) \leftarrow \frac{w_n^c(t) - w_n^{ave}(t)}{1-\lambda}, 1 \leq n \leq N.$
  end if
  Return $w_n^c(t+1)$
end for

The output of $x^{(i)}$ after batch normalization is defined as:

$$z^{(i)} = \gamma \cdot \tilde{x}^{(i)} + \beta, \tag{4.8}$$

where γ is the scaling factor and β is the shifting factor, both of which are key parameters in the training process. These variables determine whether batch normalization benefits the feature representation of the neural network model. If so, the scaling and shifting factors are updated to restore the linearly transformed output $z^{(i)}$ to the input data $x^{(i)}$. For a convolution layer with multiple channels, Equation (4.8) can be further expressed as:

$$z_j^{(i)} = \gamma_j \cdot \tilde{x}^{(i)} + \beta_j, \tag{4.9}$$

where j represents the channel index.

The proposed dynamic pruning strategy prunes unimportant parameters based on the distance between the values of the local model and the determined threshold. Since this strategy operates on the server, the average weight matrix w^{ave} serves as the threshold matrix. We also introduce the difference matrix M for each local model to record these distances, which can be calculated as follows:

$$M_n[i][j] = |w_n^c[i][j] - w^{ave}[i][j]|. \tag{4.10}$$

Then, each value in the same position of the difference matrix M_n is sorted in descending order.

After central aggregation and pruning, each local model updates its parameters through further local training to guarantee performance, following the "Training-Pruning-Fine-Tuning" strategy. The local model update is defined by the following equation:

$$w^c = argmin\,(\parallel w^c - w^{ave} \parallel^2) - \lambda \parallel w_c \parallel^2, \tag{4.11}$$

where $0 < \lambda < 1$.

According to the Lagrange multiplier method Everett III (1963), Equation (4.11) can be rewritten as:

$$\begin{aligned} L(w^c(t), u, w^{ave}(t)) = \frac{1}{2}\parallel w^c(t+1) - w^c(t) \parallel^2 - \lambda \parallel w_t^c \parallel^2 \\ -\mu^t \cdot w^c(t+1) + w_t^{ave}(1^t w^c(t+1) - 1), \end{aligned} \tag{4.12}$$

where μ^t represents the weight of dataset D on client n in the t-th round.

Supposing $((w^c(t+1))^{\cdot}\mu^{\cdot}(w^{ave}(t))^{)}$ is the optimal solution, the following Karush-Kuhn-Tucker conditions Gordon and Tibshirani (2012) must be satisfied:

$$\begin{aligned} &(w^c(t+1))^* - w^c(t) - \lambda(w^c(t+1))^* - \mu^* + (w^{ave}(t))^* = 0, \\ &1^T \cdot (w^c(t+1))^* = 1, (w^c(t+1))^* \geq 0, \mu^* \geq 0, \\ &\mu_n^*(w_n^c(t+1))^* = 0, 1 \leq n \leq N. \end{aligned} \tag{4.13}$$

If $(w_n^c(t+1))^* \geq 0$, then μ_n^* = 0. We have

$$w_n^c(t+1) = \frac{w_n^c(t) - w^{ave}(t)}{1-\lambda}. \tag{4.14}$$

Similar to Equation (4.13), if $(w_n^c(t+1))^* = 0$, then $\mu_n^* \geq 0$. We have

$$w_n^c(t) - w^g(t) \leq 0. \tag{4.15}$$

According to Equation (4.13) and (4.14), the solution of $w_n^c(t+1)$ is

$$w_n^c(t+1) = \frac{w_n^c(t) - w^{ave}(t)}{1-\lambda}, 1 \leq n \leq N. \tag{4.16}$$

4.2.3 Convergence Analysis

In this section, we establish the convergence behavior of our method. The following theorem presents the main result, which gives an upper bound of the optimality gap $F(w^g(T)) - F(w^*)$ after T iterations. All intermediate lemmas and technical proofs are deferred to Appendix A.

THEOREM 4.1. *For any $\eta \leq \frac{1}{\beta}$, the upper bound of $F(w^g(T)) - F(w^*)$ after T iterations is*

$$F(w^g(T)) - F(w^*) \leq \frac{\xi^2}{2N\rho(1+\eta\beta)(E-1)G + T\sigma(1-\frac{\beta\eta}{2})\xi^2}. \tag{4.17}$$

Theorem 4.1 establishes that the convergence of the global model depends on the number of devices N, the number of iterations T, the local update period E, and the smoothness/Lipschitz constants of the objective.

COROLLARY 4.1. *Under the conditions of Theorem 4.1, choose constants $c, \alpha \in (0, 1]$ and set the stepsize and local period as*

$$\eta \;=\; \frac{c\,N}{\sqrt{T}}, \qquad E \;=\; \left\lfloor \frac{\alpha\,T}{N^2} \right\rfloor,$$

with $\eta \leq 1/\beta$ and T large enough so that $E \geq 1$. Then the global model satisfies

$$F(w^g(T)) - F(w^*) \;\leq\; \frac{\xi^2}{c\alpha\rho G\,\sqrt{T} + \frac{c\sigma}{2}\,N\sqrt{T}} \;\leq\; \frac{2\xi^2}{c\sigma} \cdot \frac{1}{N\sqrt{T}} \tag{4.18}$$

TABLE 4.1. Comparison of convergence rate of different pruning algorithms in FL. **C.R.** represents convergence rate.

Method	Our Method	PruneFL	AAOMP	Sub-FedAvg	FedSGD	GWEP	Feddm
C.R.	$\mathcal{O}(\frac{1}{N\sqrt{T}})$	$\mathcal{O}(\frac{1}{\sqrt{NT}})$	$\mathcal{O}(\frac{1}{\sqrt{NT}})$	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\mathcal{O}(\frac{1}{\sqrt{TN}})$
Method	EFLMP	Hidenseek	DepthFL	Fedduap	FedPE	FLASH	FedMP
C.R.	$\mathcal{O}(\frac{1}{\sqrt{TN}})$	$\mathcal{O}(\frac{1}{\sqrt{TN}})$	$\mathcal{O}(\frac{1}{\sqrt{TN}})$	$\mathcal{O}(\frac{1}{\sqrt{T}})$	$\mathcal{O}(\frac{1}{\sqrt{TN}})$	$\mathcal{O}(\frac{1}{\sqrt{TN}})$	$\mathcal{O}(\frac{1}{\sqrt{T}})$

Corollary 4.1 implies that our method achieves a convergence rate of $\mathcal{O}\left(\frac{1}{N\sqrt{T}}\right)$, which matches the fastest known rate in cross-device FL (see Table 4.1). Moreover, the communication complexity of our method is $\mathcal{O}(TN^2)$.

4.3 Experiments

4.3.1 Datasets and settings

We evaluate the proposed method on six widely used benchmark datasets in FL research: MNIST Deng (2012), EMNIST Cohen *et al.* (2017), FMNIST Xiao *et al.* (2017), CIFAR-10 Recht *et al.* (2018), CIFAR-100 Xu *et al.* (2015), and CelebA Liu *et al.* (2015). These datasets were selected for two primary reasons. First, they are extensively adopted in prior work, which facilitates a fair comparison with existing approaches. Second, they span a spectrum of complexity, thereby enabling us to test the adaptability of the pruning framework. Specifically, MNIST and FMNIST provide relatively simple digit and fashion classification tasks; EMNIST extends MNIST to a larger handwritten character set (we used the Digits subset in our experiments); CIFAR-10 and CIFAR-100 introduce natural image classification tasks with increasing label granularity; and CelebA contains 202,599 face images of 10,177 individuals annotated with 40 attributes (e.g., smiling, wearing a hat), representing a large-scale, high-dimensional, and attribute-imbalanced dataset. This progression from simple to complex datasets allows us to evaluate whether pruning remains effective under increasing data difficulty and heterogeneity.

To further investigate model generality, we employed four representative architectures as local models on each client: a Multi-layer Perceptron (MLP), a Convolutional Neural Network

TABLE 4.2. Setting of hyper-parameters and data sets

	MNIST-CNN	EMNIST-CNN	FMNIST-CNN
Number of labels	10	10	10
Number of training samples	5500	24000	60000
Number of testing samples	1000	4000	10000
Learning rate η	0.02	0.02	0.02
Local batch_size B	32	32	32
Communication rounds T	100	100	100
Local epoch E	10	10	10
	CIFAR10-VGG9	CIFAR100-ResNet18	Celeba-VGG9
Number of labels	10	10	40
Number of training samples	50000	50000	141819
Number of testing samples	10000	10000	60780
Learning rate η	0.01	0.01	0.01
Local batch_size B	128	128	64
Communication rounds T	400	400	200
Local epoch E	20	20	20

(CNN), a lightweight VGG9, and a ResNet18. The MLP and CNN represent lightweight models commonly used in FL, while VGG9 and ResNet18 capture deeper and more expressive models suitable for more complex vision tasks. Hyper-parameter settings and model architectures are provided in Table 4.2 and Table 4.3, respectively.

For training, we used stochastic gradient descent (SGD) for local optimization and the standard FedAvg algorithm for aggregation on the server. In each communication round, 10 clients were randomly selected to participate. To simulate statistical heterogeneity, we adopted a non-IID setting generated by partitioning data according to a Dirichlet distribution Lin *et al.* (2020), where the concentration parameter a controls the level of skew (smaller values indicate stronger non-IID effects). This allows us to systematically study the robustness of pruning under varying degrees of heterogeneity.

All experiments were conducted on a server equipped with an NVIDIA GeForce RTX 3060 GPU, Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz, and 12GB of VRAM. To better mimic a realistic FL environment, aggregation and pruning operations were executed on the CPU, while local neural network training was performed on the GPU.

TABLE 4.3. Architecture of different local models.

Architecture	Convolutional layer	Full connected layer
CNN	32, pool 64, pool	2048,10 (input: 2048)
VGG9	$32, 64$, pool 2×128, pool 2×256, pool	$256, 256, 10$ (input:256)
ResNet18	64, pool $2 \times [64, 64]$, $2 \times [128, 128]$, $2 \times [256, 256]$, $2 \times [512, 512]$	avgpool, 100 (input:512)

4.3.2 Evaluation metrics

To comprehensively assess the effectiveness of the proposed pruning framework, we report results across multiple evaluation metrics commonly used in FL studies.

Test accuracy. Model performance is primarily measured by the classification accuracy on the global test set. Accuracy directly reflects whether pruning preserves the representational capacity of the model after reducing communication costs.

Communication cost. Since pruning is introduced as a communication-reduction mechanism, we measure the number of transmitted parameters in each communication round and the cumulative transmission cost until convergence. This metric captures the efficiency gains provided by pruning.

Convergence speed. We further track the number of communication rounds required to reach a target accuracy (e.g., 90% on MNIST or 70% on CIFAR-10). This allows us to evaluate whether pruning accelerates convergence by mitigating communication bottlenecks.

Robustness under heterogeneity. To investigate stability, we evaluate pruning performance under varying levels of non-IID data distributions induced by the Dirichlet parameter a. We also compare across different model architectures to assess whether pruning maintains effectiveness as task and model complexity increase.

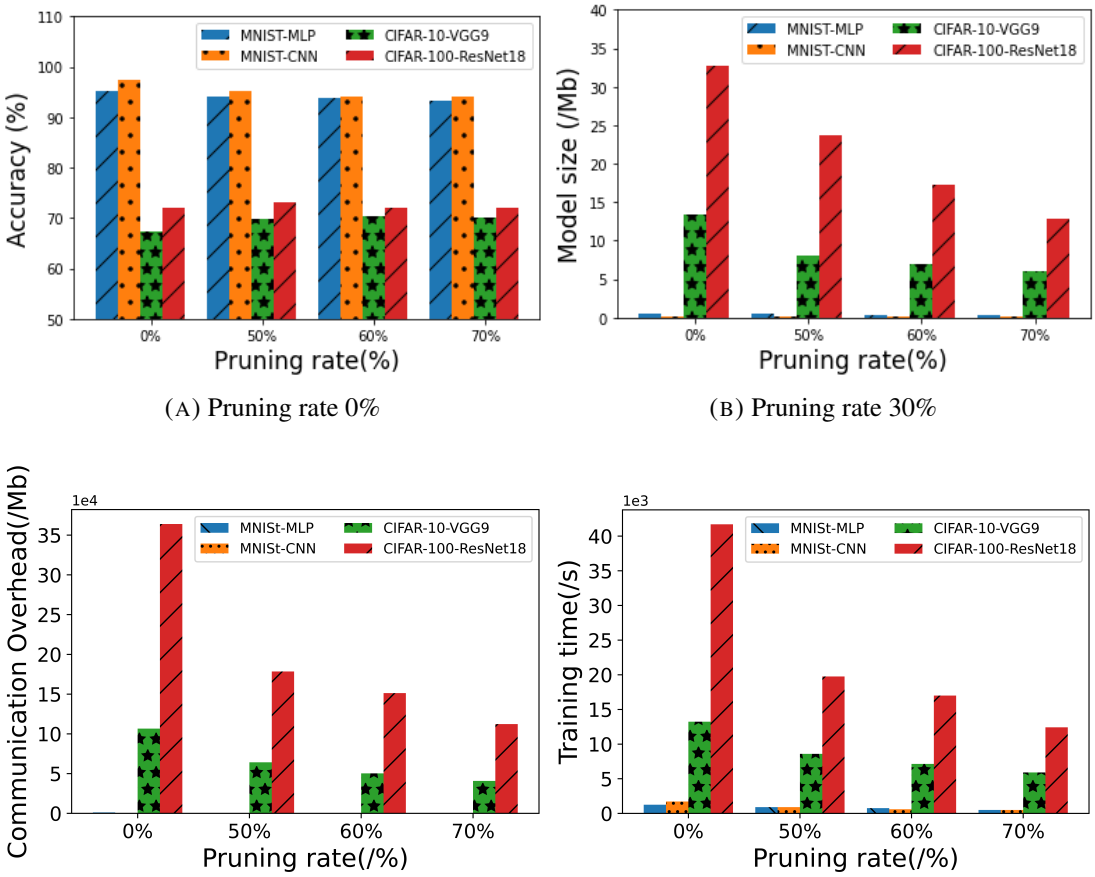
4.3.3 Baselines

We compared our pruning method with some baseline and benckmark algorithms (from 2018 to 2024); namely, FedAvg McMahan *et al.* (2017), FedFusion Duan *et al.* (2023), FeddmXiong *et al.* (2023), FL-PQSU Xu *et al.* (2021b), PrunFLJiang *et al.* (2022a), GWEP Prakash *et al.* (2022) , EFLMP Wu *et al.* (2023b), HidenseekVallapuram *et al.* (2022), DepthFLKim *et al.* (2022), FedduapZhang *et al.* (2022a), FedPEYi *et al.* (2024), FLASH-and-PruneSalehi *et al.* (2024), FedMP Jiang *et al.* (2023), PHFL Pervej *et al.* (2024) and Sub-FedAvg Vahidian *et al.* (2021). Since FedAvg, FedFusion, and Feddm algorithms are not adapted to the pruning rates, we set the pruning rate as 60% for a fair comparison.

Figure 4.1 shows the performance of our method on the MNIST, CIFAR-10, and CIFAR-100 datasets. We used model size, communication overhead, and training time to evaluate different aspects of communication cost. The results of our pruning method align with and support the theoretical findings discussed earlier. Notably, the accuracy of our method remains highly stable, with no more than a 3% variation, even as the pruning rate increases from 0% to 70%. While neural network pruning did not significantly impact model accuracy, communication overhead was reduced by more than 50% across all datasets. For instance, on the CIFAR-10 dataset, communication overhead was reduced by approximately 55%, and on the CIFAR-100 dataset, it was reduced by nearly 60%.

4.3.4 Results

The training time also reflects the impact of our pruning method, especially as neural network architectures become deeper and datasets larger. For example, when using the ResNet18 network trained on CIFAR-100, training time was halved after pruning. For smaller datasets like MNIST, the reduction in training time was less dramatic but still achieved a 30% improvement. These experimental results confirm the effectiveness of our pruning method in reducing communication overhead and training time while maintaining high model accuracy. The stability of accuracy, even with significant pruning, highlights the robustness of our approach.



(A) Pruning rate 0%

(B) Pruning rate 30%

(C) Pruning rate 50%

(D) Pruning rate 70%

FIGURE 4.1. Performance evaluation of our method on different datasets with varying pruning rates.

More detailed discussions and additional experimental results can be found in the **Discussion** section, which further elaborates on the practical benefits and implementation of our method.

4.3.5 Overall comparison

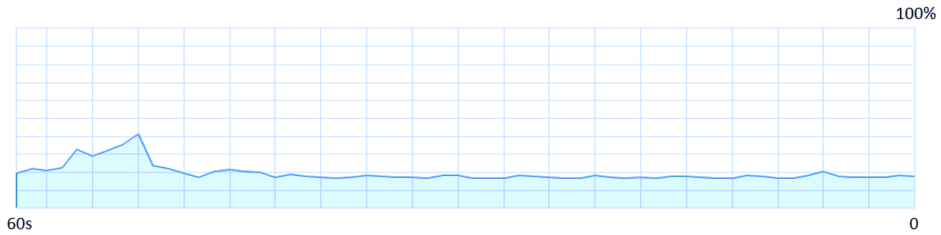
To comprehensively evaluate the performance of our method, we compared it with 15 state-of-the-art methods on six popular image datasets. Since the pruning rate is a hyperparameter in both our method and several others, we set the pruning rate to 60% for a fair comparison. As shown in Table 4.6, our method outperformed the baselines in nearly all tests in terms of accuracy. Additionally, our method demonstrated lower communication costs across all datasets compared to the baselines, reflecting its effectiveness in removing more redundant

parameters while maintaining accuracy. As the number of neural network parameters decreased, the training time for all methods was reduced, but our method's training time was significantly lower than that of the baselines in all tests. As mentioned in the introduction, methods such as EFLMP, Hidenseek, DepthFL, and FedDUAP did not outperform certain client-side pruning methods, which highlights a known issue where traditional server-side pruning sometimes falls short compared to client-side pruning. However, we believe that server-side pruning has inherent advantages due to its ability to leverage global knowledge and centralized processing power. Our proposed server-side pruning method addresses the drawbacks of existing server-side approaches by including a critical restore step post-pruning. This ensures that any important parameters removed during pruning are recovered, preserving overall model accuracy. This key improvement allows our server-side pruning algorithm to outperform both client-side and traditional server-side methods. In summary, although some server-side pruning methods may not perform as well as client-side methods, our proposed server-side pruning algorithm with a restore step clearly demonstrates superior performance. Our experimental results confirm the advantages of server-side pruning, and we are confident in its effectiveness.

4.3.6 Computation consumption

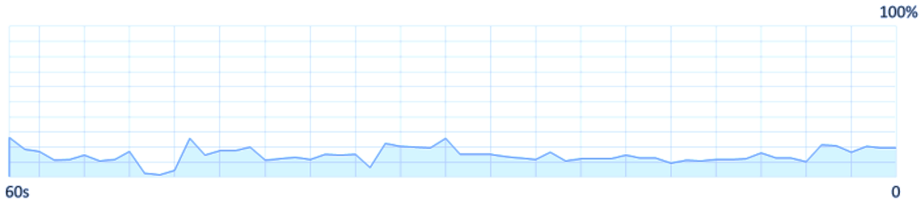
We evaluated the GPU and CPU utilization by running the neural network models 20 times across 100 communication rounds to investigate the computational demands of two pruning methods. Figure 4.2 illustrates the CPU and GPU utilization for our server-side pruning approach compared to the traditional local pruning method, PrunFL, during a random communication round. PrunFL is selected as a representative client-side pruning baseline, allowing us to highlight the system-level differences between client-side and server-side pruning strategies. Table 4.4 provides an overall view, showing average CPU and GPU utilization, and reveals a decrease in GPU utilization with a corresponding increase in CPU utilization when pruning shifts to the server. Our results also demonstrate that server-side pruning enhances system efficiency with respect to pipeline latency. As detailed in Table 4.5, we measured the average

duration of each key operation in FL and observed that our method requires twice the aggregation time. However, the total pipeline latency of our approach is significantly lower than that of PrunFL. This increased efficiency is attributed to the server's advanced computational capability, enabling faster pruning operations compared to the GPU. As a result, server-side pruning leads to improved efficiency, optimal resource allocation, and contributes to enhanced system performance and reliability.



(A) CPU utilization of ours

(B) GPU utilization of ours



(C) CPU utilization of PrunFL

(D) GPU utilization of PrunFL

FIGURE 4.2. CPU and GPU utilization comparison between two types pruning methods.

TABLE 4.4. Average utilization of GPU and CPU by running the neural network models 20 times across 100 communication rounds.

	Model	CPU (%)	GPU (%)
Our method	MLP	25.81	41.55
	VGG9	30.14	53.24
PrunFL	MLP	12.18	65.91
	VGG9	13.87	75.62

TABLE 4.5. Pipeline latency of two pruning methods.

	Our method		PrunFL	
	MLP	VGG9	MLP	VGG9
Training (/s)	5.14	14.24	5.22	15.09
Pruning (/s)	3.14	8.21	3.54	11.25
Aggregation (/s)	1.91	5.24	1.01	3.24
Pipeline latency (/s)	513.91	1423.21	521.33	1508.4

4.3.7 Training loss

Training loss directly reflects the convergence performance of different methods. As shown in Figure 4.3, our method had a faster convergence rate than the other methods on the larger datasets. Since our method can quickly identify and remove redundant parameters on the server, it converged within 50 rounds. In contrast, the baselines were noticeably slower in converging, as they are client-side pruning solutions that require extra communication rounds to determine the crucial parameters for the global model, which are pruned locally. Additionally, the convergence curves of other methods, such as GWEP and Sub-FedAvg, began to fluctuate in the later stages of training.

4.3.8 Minimum communication rounds

We tested the minimum number of communication rounds required for different pruning methods to achieve 90% accuracy on the MNIST and EMNIST datasets, 75% on the CelebA and FMNIST datasets, and 70% on the CIFAR-10 and CIFAR-100 datasets. As shown in Figure 4.4, our method required the fewest communication rounds to achieve the target accuracy across all datasets. This result suggests that the difference in communication rounds is inherent to the method. The baselines could only adjust their parameters through

TABLE 4.6. The results of the performance evaluations of all pruning methods on six data sets.

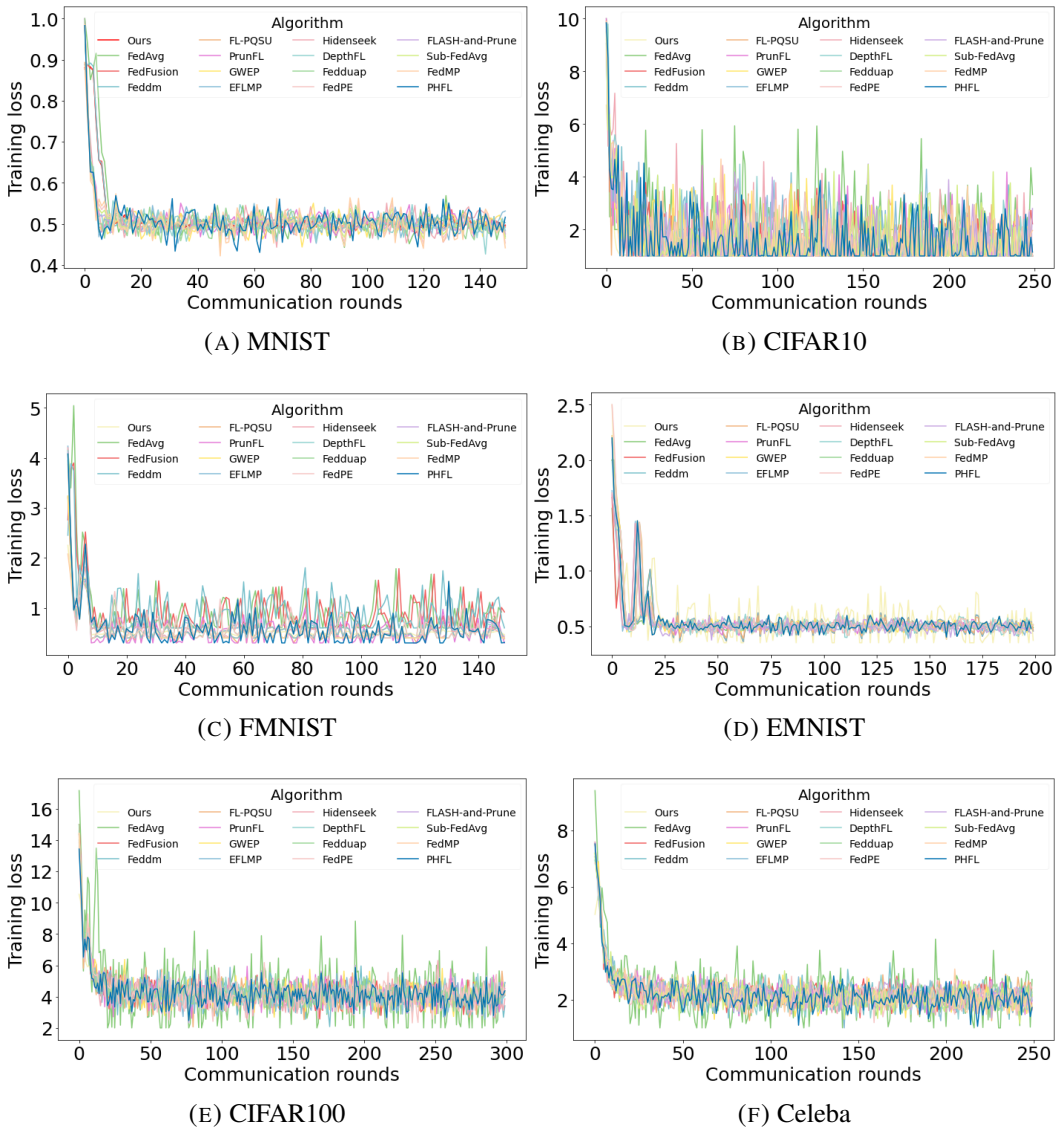
Accuracy(/%)	**Ours**	FedAvg	FedFusion	Feddm	FL-PQSU	PrunFL	GWEP	EFLMP
MNIST	**93.92**	86.83	89.25	82.52	89.34	90.12	92.21	90.15
EMNIST	**92.12**	82.14	86.24	79.54	87.41	89.15	90.45	88.54
FMNIST	**90.44**	83.45	82.48	75.45	83.75	87.45	88.54	86.71
Celeba	**88.45**	72.45	60.45	68.39	75.62	73.12	76.15	72.64
CIFAR10	**70.24**	61.25	59.31	51.24	69.31	68.78	71.75	69.45
CIFAR00	**73.42**	55.87	53.41	49.51	66.78	66.78	70.21	65.91
Accuracy(/%)	Hidenseek	DepthFL	Fedduap	FedPE	FLASH-and-Prune	Sub-FedAvg	FedMP	PHFL
MNIST	89.15	90.05	89.97	89.87	88.81	90.11	90.11	90.51
EMNIST	87.51	88.01	88.41	88.54	87.64	89.34	88.14	88.64
FMNIST	87.01	86.94	87.91	86.91	87.04	88.45	87.91	86.47
Celeba	73.45	73.54	74.51	75.04	76.04	82.15	81.01	79.51
CIFAR10	70.04	70.45	71.41	72.01	73.14	70.75	75.14	71.45
CIFAR00	66.14	66.91	66.45	67.54	68.45	72.14	72.41	72.14
Overhead(/MB)	**Ours**	FedAvg	FedFusion	Feddm	FL-PQSU	PrunFL	GWEP	EFLMP
MNIST	**8.41E+01**	1.28E+02	9.04E+01	9.01E+01	1.10E+02	8.82E+01	9.02E+01	9.45E+01
EMNIST	**9.08E+01**	1.46E+02	1.20E+02	1.19E+02	1.30E+02	9.02E+01	1.20E+02	1.05E+02
FMNIST	**9.28E+01**	1.00E+02	9.84E+01	9.92E+01	1.09E+02	9.55E+01	1.12E+02	1.08E+02
Celeba	**1.00E+05**	1.65E+05	1.42E+05	1.50E+05	1.85E+05	1.00E+05	1.86E+05	1.65E+05
CIFAR10	**5.02E+04**	7.70E+04	5.42E+04	5.41E+04	6.62E+04	6.54E+04	6.45E+04	6.15E+04
CIFAR100	**1.51E+05**	2.31E+05	1.62E+05	1.62E+05	1.98E+05	1.76E+05	1.66E+05	1.75E+04
Overhead(/MB)	Hidenseek	DepthFL	Fedduap	FedPE	FLASH-and-Prune	Sub-FedAvg	FedMP	PHFL
MNIST	9.30E+01	9.40E+01	9.35E+01	9.28E+01	9.33E+01	8.95E+01	9.43E+01	9.23E+01
EMNIST	1.00E+02	1.02E+02	1.00E+02	9.95E+01	1.00E+02	1.15E+02	1.03E+02	9.90E+01
FMNIST	1.06E+02	1.11E+02	1.08E+02	1.07E+02	1.08E+02	1.11E+02	1.13E+02	9.98E+02
Celeba	1.70E+05	1.75E+05	1.70E+05	1.65E+05	1.68E+05	1.79E+05	1.75E+05	1.60E+05
CIFAR10	6.09E+04	5.95E+04	6.04E+04	6.04E+04	5.99E+04	6.79E+04	6.81E+04	6.05E+04
CIFAR100	1.68E+04	1.70E+04	1.69E+04	1.66E+04	1.65E+04	1.67E+05	1.66E+04	1.60E+04
Training time(/s)	**Ours**	FedAvg	FedFusion	Feddm	FL-PQSU	PrunFL	GWEP	EFLMP
MNIST	**6.04E+02**	1.83E+03	1.65E+03	1.58E+03	1.00E+03	1.76E+03	9.95E+02	1.62E+03
EMNIST	**8.00E+02**	2.10E+03	1.95E+03	1.87E+03	1.22E+03	2.05E+03	1.10E+03	1.59E+03
FMNIST	**9.15E+02**	1.65E+03	1.57E+03	1.48E+03	1.35E+03	1.68E+03	9.06E+02	1.12E+03
Celeba	**1.21E+05**	4.52E+06	5.05E+06	4.86E+06	1.68E+06	5.22E+06	1.59E+06	1.67E+06
CIFAR10	**7.13E+03**	2.17E+04	1.95E+04	2.00E+04	8.91E+03	2.68E+04	7.91E+04	5.75E+04
CIFAR100	**1.70E+04**	6.15E+04	5.65E+04	4.45E+04	2.31E+04	6.96E+04	1.80E+05	8.10E+04
Training time(/s)	Hidenseek	DepthFL	Fedduap	FedPE	FLASH-and-Prune	Sub-FedAvg	FedMP	PHFL
MNIST	1.62E+03	1.65E+03	1.61E+03	1.62E+03	1.61E+03	9.86E+02	1.71E+03	1.71E+03
EMNIST	1.60E+03	1.62E+03	1.63E+03	1.60E+03	1.53E+03	1.10E+03	1.54E+03	1.52E+03
FMNIST	1.10E+03	1.10E+03	1.20E+03	1.10E+03	1.07E+03	9.10E+02	1.46E+03	1.14E+03
Celeba	1.47E+06	1.56E+06	1.61E+06	1.72E+06	1.69E+06	1.57E+06	1.68E+06	1.71E+06
CIFAR10	5.68E+04	5.77E+04	6.05E+04	6.15E+04	7.05E+04	7.42E+04	6.98E+04	7.02E+04
CIFAR100	8.09E+04	8.12E+04	7.99E+04	7.97E+04	7.97E+04	1.80E+05	8.01E+04	7.99E+04

a=0.2. We take the average value from five times repeat.

communication with the server, whereas our method adjusted the parameters in each pruning iteration on the server, resulting in fewer communication rounds.

4.3.9 Non-IID

Pruning more than 50% of a neural network's parameters often impacts performance. To test robustness, we evaluated multiple methods on two datasets under different levels of non-IID



(A) MNIST (B) CIFAR10

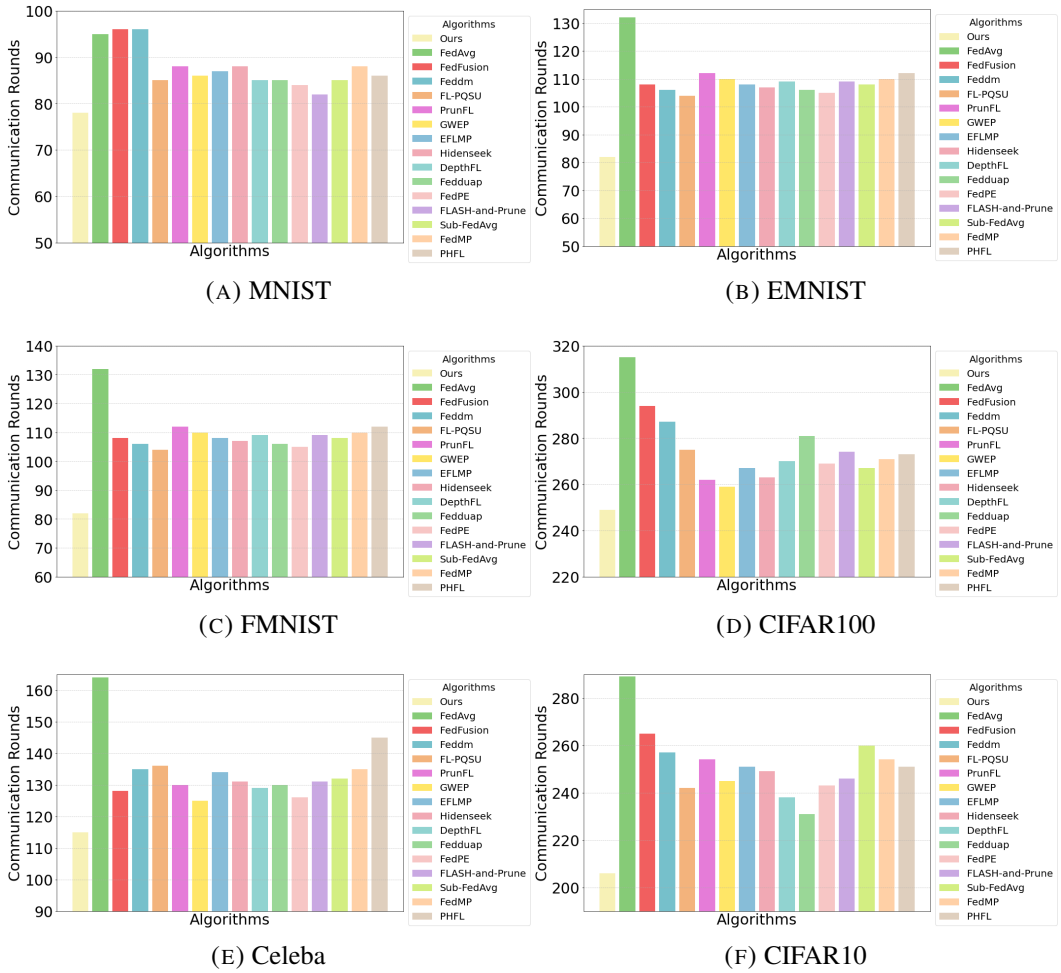
(C) FMNIST (D) EMNIST

(E) CIFAR100 (F) Celeba

FIGURE 4.3. Training loss of different methods on six data sets.

data distribution. As the non-IID level increased, baseline methods saw sharp accuracy drops on CIFAR-100, while our method remained stable (Figure 4.5). Our server-side pruning leveraged global knowledge to adjust parameters, while baseline methods struggled as local pruning removed crucial parameters for global aggregation.

Our method preserves key global parameters by evaluating the similarity between local and global models, selectively pruning less important ones. This ensures model performance remains stable across varying non-IID data distributions, whereas baseline methods fail to



(A) MNIST (B) EMNIST

(C) FMNIST (D) CIFAR100

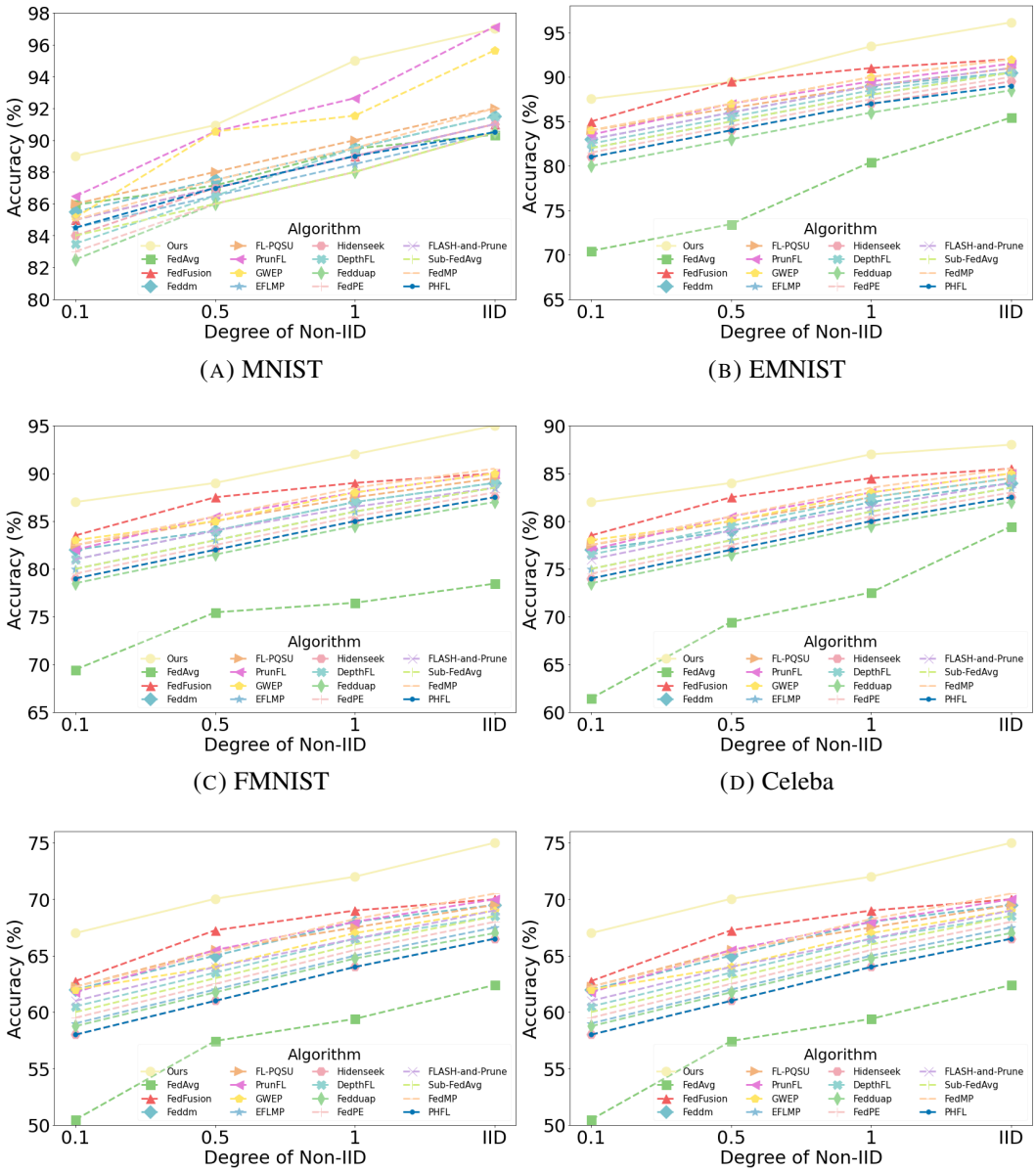
(E) Celeba (F) CIFAR10

FIGURE 4.4. Minimum communication rounds of different methods under different degree of Non-IID.

recover removed parameters. By identifying critical parameters for global generalization and reducing model complexity, our approach minimizes communication overhead. To mitigate performance loss, we employ a recovery step, such as fine-tuning, to maintain model accuracy and robustness.

4.4 Discussion

To investigate the influence of pruning rate on the training progress, we compared the convergence performance at increasing pruning rates using the same local model and datasets.



(A) MNIST (B) EMNIST

(C) FMNIST (D) Celeba

(E) CIFAR100 (F) CIFAR10

FIGURE 4.5. Accuracy of different methods under different degree of Non-IID.

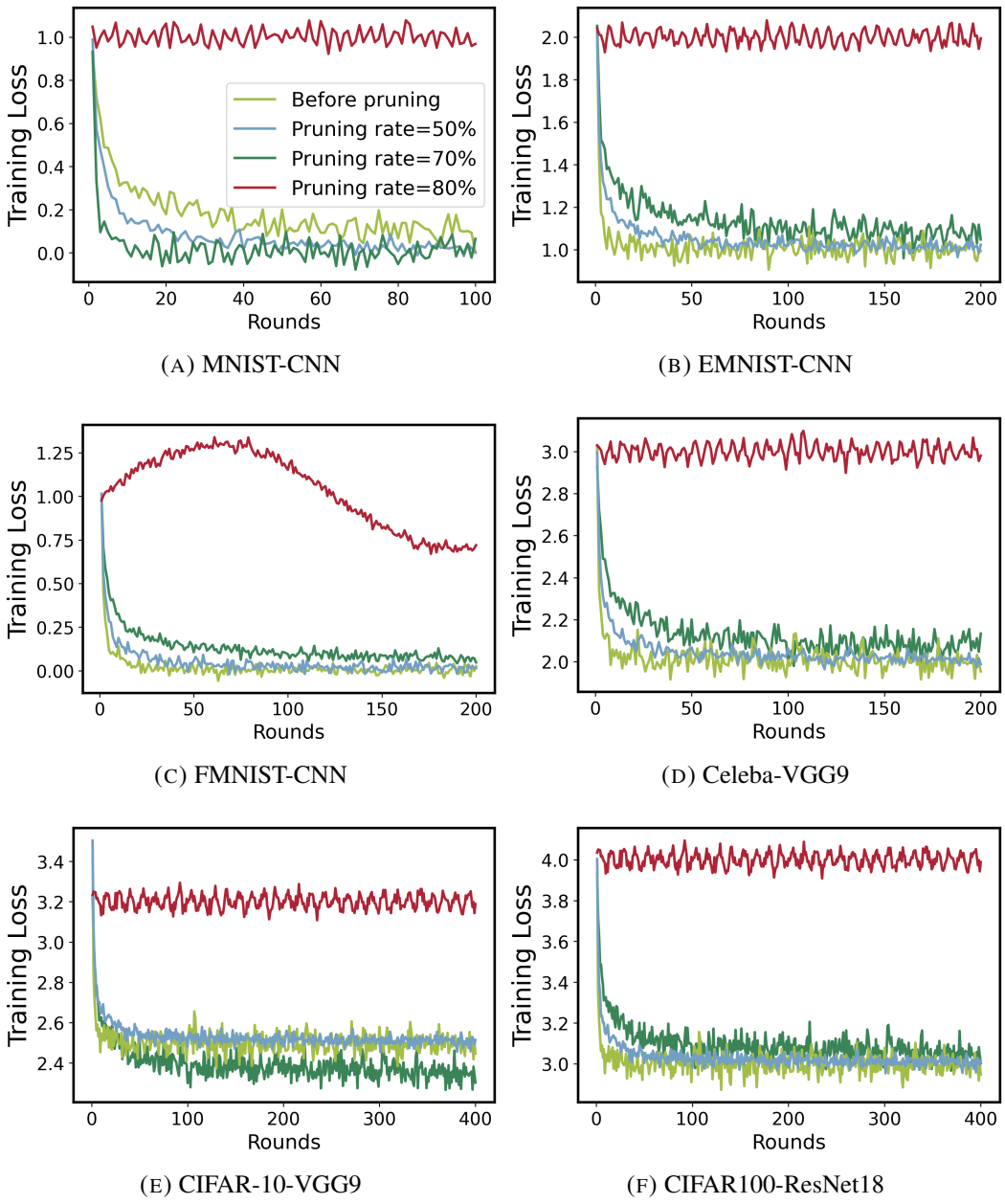
Figure 4.6 shows the convergence progress of six cases under a federated training environment with different pruning rates on various datasets. In Figures 4.6 (a), (b), and (c), the convergence curve was smooth, converging by the fifth round before pruning. When we increased the pruning rate to 50% and 70%, the training loss gradually increased, and more communication rounds were required for convergence. At an 80% pruning rate, the CNN model failed to

converge. This suggests that if too many parameters are removed at once, local training rounds are insufficient to recover the damaged neural network structure. However, in Figures 4.6 (d), (e), and (f), the opposite effect was observed. The training loss of the original VGG9 and ResNet18 models was 70% greater than that of the pruned models. As the pruning rate increased from 50% to 70%, the curves converged earlier. Notably, when the pruning rate reached 80%, the models could no longer converge. This result suggests that approximately 30% of the parameters play a crucial role in the convergence of neural network models.

To investigate the impact of the number of participating clients on the performance of our method, we measured the test accuracy, communication overhead, and training time with different ratios of participating clients (10, 20, 50, 100) across six datasets. As shown in Table 4.7, the accuracy exhibited a slight downward trend as the number of clients increased, but there was no significant drop. For example, the accuracy on the MNIST dataset decreased from 93.92% with 10 clients to 90.14% with 100 clients, and on the CIFAR-100 dataset, it dropped from 73.42% to 70.94%. This decrease in accuracy remains within an acceptable range, with the model still maintaining high performance. Communication overhead, however, increased significantly with the number of clients. On the CelebA dataset, for instance, communication overhead rose from around 101,000MB with 10 clients to around 989,000MB with 100 clients. This growth is expected, as more clients result in more data being transferred between clients and servers. Training time also increased as the number of clients grew. For example, on the FMNIST dataset, training time rose from 915 seconds with 10 clients to 2,650 seconds with 100 clients. The increase in training time reflects the added computational and synchronization complexity when more clients participate in training. As the number of clients grows, the frequency of data processing and global model updates increases, making the training process more time-consuming.

4.5 Conclusion

In this chapter, we proposed a server-side adaptive pruning method that leverages the similarity between global and local models to guide parameter pruning. Within the unified problem



(A) MNIST-CNN (B) EMNIST-CNN

(C) FMNIST-CNN (D) Celeba-VGG9

(E) CIFAR-10-VGG9 (F) CIFAR100-ResNet18

FIGURE 4.6. Training loss of our method on different data sets under different pruning rates. The legend is the same in both sub-figures and 'MNIST-CNN' means the CNN model is as the local model on MNIST data set.

definition, this method directly addresses the communication overhead term $\mathcal{C}(w)$. From a theoretical perspective, we established that the proposed approach achieves the best-known convergence rate under the FL setting. Extensive experiments further demonstrate that it accelerates convergence and preserves accuracy in non-IID scenarios, while reducing

TABLE 4.7. The impact of number of participating clients on the performance of our method on different data sets. The heterogeneous environment is created by the Dirichlet distribution and the non-IID degree is set as 0.2.

Dataset	Number of clients	Accuracy(%)	Com(/MB)	Training time (/s)
MNIST	10	93.92	8.41E+01	6.04E+02
	20	91.24	1.69E+02	8.14E+02
	50	90.67	3.97E+02	1.45E+03
	100	90.14	8.03E+02	1.97E+03
CelebA	10	88.45	1.01E+05	1.21E+05
	20	86.54	1.99E+05	1.54E+05
	50	84.61	5.10E+05	1.98E+05
	100	82.15	9.89E+05	2.01E+05
EMNIST	10	92.12	9.08E+01	8.00E+02
	20	90.14	1.78E+02	1.54E+03
	50	87.64	4.34E+02	1.94E+03
	100	85.45	8.75E+02	2.35E+04
CIFAR10	10	70.24	5.02E+04	7.13E+03
	20	68.54	1.14E+05	9.38E+03
	50	67.41	2.34E+05	1.24E+04
	100	67.01	4.97E+05	1.54E+04
FMNIST	10	90.44	9.28E+01	9.15E+02
	20	89.25	1.89E+02	1.64E+03
	50	87.61	4.35E+02	2.03E+03
	100	84.99	9.14E+02	2.65E+03
CIFAR100	10	73.42	1.51E+05	1.70E+04
	20	72.41	2.98E+05	1.98E+04
	50	71.62	5.09E+05	2.21E+04
	100	70.94	1.48E+06	2.61E+04

communication volume by 50%–70% across training rounds. By alleviating the fundamental bottleneck of parameter transmission, this chapter validates the effectiveness and feasibility of neural network pruning for improving communication efficiency in FL, and provides methodological support for its application in more complex communication environments.

CHAPTER 5

Scaling Up Neural Network Pruning for Large-Scale Federated Learning Systems

---

In the previous chapter, we demonstrated that neural network pruning can effectively alleviate the communication cost in FL base systems. Compared with the server-side pruning strategy introduced in the previous chapter, large-scale FL environments introduce additional design challenges. In particular, communication efficiency must now be considered in both the upload and download phases, and pruning decisions must remain consistent across a much larger population of heterogeneous clients. In this chapter, we further investigate how neural network pruning can be applied to improve communication efficiency when the number of participating clients increases dramatically. A typical and practical application scenario is intelligent transportation systems (ITS), where a large number of vehicles participate in training as clients. Tasks such as autonomous driving and vehicle detection require large-scale models to process diverse and dynamic data streams. When hundreds or thousands of vehicles participate in training simultaneously, uploading local model parameters and downloading the global model incur significant communication overhead, directly reducing the overall efficiency of FL.

5.1 Introduction

As FL scales to large-scale deployments, the number of participating clients increases substantially, leading to significant communication challenges. Each additional client introduces extra upload and download traffic, and in aggregate, this rapidly amplifies the overall communication cost required to train a global model. Such real world applications like ITS,

where hundreds or even thousands of vehicles or edge devices may participate in collaborative learning tasks Zhang *et al.* (2023). In these scenarios, FL is attractive because it enables multiple clients to train shared models while preserving data privacy Jin *et al.* (2023), but the sheer scale of participation magnifies the communication burden and creates an urgent need for more efficient solutions.

One of the primary challenges in such larger-client FL environments is the sharp increase in communication burden caused by the combination of numerous participating clients and excessively large model sizes. In each training round, clients must upload and download model updates; when the number of clients grows, even relatively small updates accumulate into a massive volume of traffic. This problem is further exacerbated by the use of deep neural networks with millions of parameters, where transmitting model updates consumes substantial bandwidth and may incur high communication costs on mobile or resource-constrained networks Liu *et al.* (2022); Zhou *et al.* (2023). Moreover, transmission delays caused by network fluctuations or limited bandwidth undermine the real-time requirements of large-scale systems, making communication efficiency a critical bottleneck for scalable FL.

While structured pruning methods have successfully reduced upload communication volume Huang *et al.* (2023); Jiang *et al.* (2022a,b), they are primarily applied to prune local networks on the client side, leading to two significant challenges. **First**, local network pruning often relies on the characteristics of local datasets, which means that structures considered redundant locally might still be crucial for the global model. Consequently, optimizing the local model's performance could diminish the effectiveness of the global model Molchanov *et al.* (2019). **Second**, while local model pruning effectively reduces upload volume, it typically has minimal impact on download volume. In the heterogeneous environment of FL, different clients may prune various neurons or filters, leading to negligible reductions in the global model's size Caldas *et al.* (2018). Filters generally extract features from input data, and "neurons" are the basic computational units of the network. As illustrated in Figure 5.1, the second neuron in (a) and the third neuron in (b) of the second layer are pruned. According to federated aggregation rules, the global model's second layer would then consist of the second neuron from (b) and the third neuron from (a). Despite local pruning efforts, the global model's size

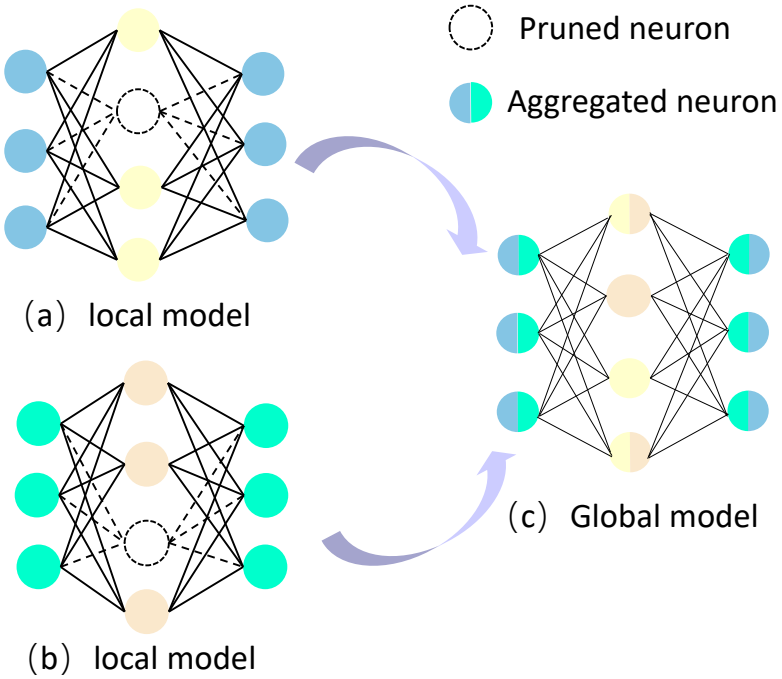


FIGURE 5.1. Illustration of Local Pruning Impact on Global Model in FL.

remains unchanged, revealing a fundamental limitation in current pruning strategies within FL environments.

Current FL pruning algorithms, particularly one-shot pruning, initially reduce upload communication volume but fail to significantly decrease the network model size in subsequent communications aimed at restoring performance. Dynamic pruning algorithms, although they ensure reduced communication volume per round, do not consistently guarantee satisfactory performance across all clients. Therefore, there is a pressing need for a pruning method that minimizes parameter involvement in both communication and aggregation, while still maintaining robust model performance.

To address the above issues, it is necessary to carefully select the key parameters involved in communication and aggregation, ensuring that the number of transmitted parameters is minimized while their contribution to the global model is maximized. However, designing such a pruning strategy in large-scale FL environments introduces several new challenges.

Compared with the server-side pruning framework proposed in Chapter 4, the large-scale client setting considered here requires communication efficiency to be optimized in both the upload and download phases. When the number of participating clients increases dramatically, reducing only the upload communication cost is no longer sufficient, since the global model must also be repeatedly distributed to a large population of clients.

In addition, pruning decisions must balance the importance of parameters from both local and global perspectives. Parameters that appear redundant for a single client may still be important for the global model, making it challenging to design a pruning strategy that preserves global knowledge while reducing communication overhead.

We propose the Federated Client and Global (FEDCG) pruning method, a bifurcated approach tailored for FL environments. FEDCG consists of two stages. 1) At the client level, FEDCG assesses each neuron or filter's mutual information with the final output to evaluate their importance. Neurons or filters with high mutual information significantly influence the final output, while those with low mutual information are considered redundant and pruned. This helps simplify the model without substantially affecting performance.

2) After receiving locally pruned models, the server conducts a comprehensive analysis of the mutual information across all filters or neurons from the participating models, retaining those with higher values. This phase involves progressively increasing pruning intensity in successive communication rounds, ensuring that essential features are retained and redundant ones are discarded. Distinct from existing one-shot pruning algorithms, FEDCG not only curtails the volume of uploaded data but also strategically reduces the download communication volume through meticulous server-side pruning of the global model. Moreover, in contrast to dynamic pruning algorithms, FEDCG adaptively balances the reduction in communication volume with ongoing adjustments in pruning intensity, thereby ensuring sustained model performance. This innovative method marks a significant stride in FL, adaptively addressing the dual challenges of communication efficiency and model efficacy within a distributed learning framework.

Our main contributions are summarized as follows:

- We are the **first** to propose a novel two-stage pruning strategy at both the client and server levels in the FL domain, which effectively reduces communication overheads in both the uploading and downloading phases.
- We introduce a novel method that utilizes mutual information to assess the importance of individual neurons or filters within a neural network. Unlike conventional performance metrics, the proposed approach delves into the complex interactions among various model components, examining their combined influence on the ultimate output.
- Extensive experiments on multiple datasets have validated the superiority of FEDCG in reducing communication costs while maintaining high accuracy, which surpasses the existing methods by a large margin, and makes it a valuable tool in ITS.

5.2 Methodology

5.2.1 Overview of our FL framework

In this section, we describe the overview of our pruning method, as illustrated in Figure 5.2. Our method is general and applicable to any deep learning model, consisting of two stages: local pruning on the client side and global pruning on the server side. Each well-trained client network records neuron or filter activation values in response to inputs, reflecting their responsiveness. The client computes the mutual information between neuron or filter outputs and the model's final outputs to assess their contribution. Neurons or filters with higher mutual information are retained, while redundant ones with lower values are removed to reduce model size and communication cost. After local pruning, the server receives pruned models and mutual information values, aggregating them into a global model by selecting high-information neurons or filters from each layer. This preserves the most important features while optimizing performance and communication efficiency. The server may then apply further pruning to simplify the model without compromising accuracy or generalization.

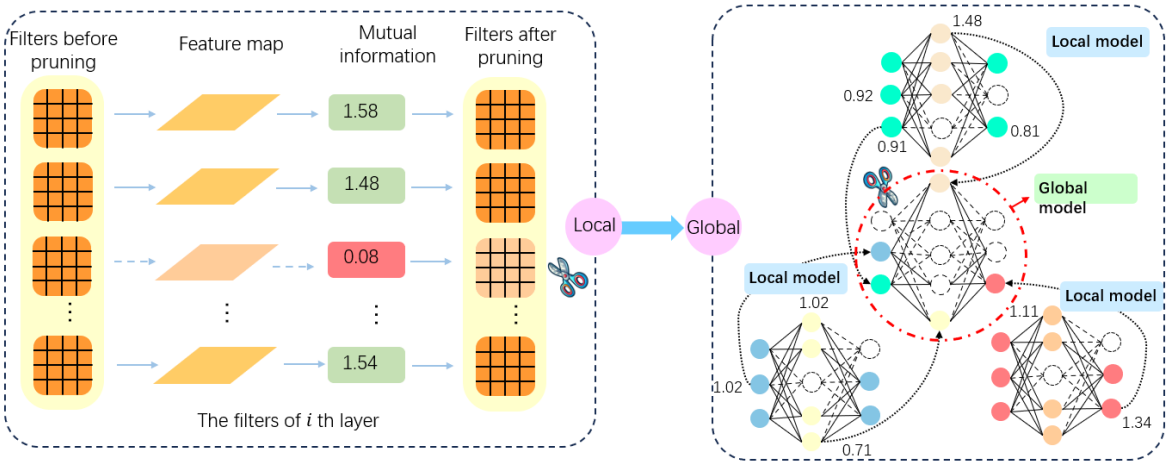


FIGURE 5.2. Overview of our **FEDCG** method. Different from existing methods, Our **FEDCG** method prunes both the local and global model simultaneously with the value of mutual information. Assuming a pruning rate of 60%, each global neuron is determined by selecting the maximum mutual information value from the three local models.

5.2.2 Problem statement

The aim of neural network pruning in our study is to reduce the upload overhead via local pruning and the download overhead via global pruning. Give N clients with data sets $D_n = \{(x^i), y^{(i)}\}$, the neural network model W_n with L layers and K neurons or filters in layer l. For local pruning, the expected risk function usually evaluates the performance of the model, which is defined as follows:

$$R(W) = E_{(x^{(i)},y^{(i)})\sim p_r}[\mathcal{L}(y^{(i)}, f(x^i; W))], \tag{5.1}$$

where $(x^{(i)}, y^{(i)}) \sim p_r()$ denotes that data point $(x^{(i)}, y^{(i)})$ follows the data distribution $p_r()$ and $\mathcal{L}(y^{(i)}, f(x^i; W))$ is the loss function.

Since the data distribution is unknown, the empirical risk function $\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y^{(i)}, f(x^i; W))$ approximates the expected risk function. Therefore, the objective function of local pruning can be described as follows:

$$\min_{W\in\mathbb{R}^d} \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y^{(n)}, f(x^{(n)}; W)) + \frac{\mu}{2}\|W\|^2 + \eta\sum_{l=1}^{L}\|W_l\|_0, \tag{5.2}$$

where $\|W\|^2$ is L_2-norm, which is employed to prevent overfitting and enhance the generalization capability of the model, and $\|W_l\|_0$ is the L_0-norm, utilized for model sparsification. The L2-norm adds a penalty term to the loss function proportional to the sum of the squared weights. This technique discourages the model from learning excessively large weights, which can lead to overfitting by ensuring that the weights remain small and distributed more evenly across the network. By applying L2 regularization during the training and pruning process, we ensure that the model generalizes well to unseen data while maintaining its sparsity. To solve this function and find the optimal W, we define $f(W_L; x) = \sigma_L(W_L\sigma_{L-1}(W_{L-1}...\sigma_1(W_1x)))$, which can be easily verified by the differential mean value theorem to be a gradient Lipsitz continuous function restricted to compact sets Goldstein (1977). For a L layers neural network, the set of number of filters or neurons is $\{M_1, M_2, ..., M_L\}$. The optimal set after pruning is $\{M_1', M_2', ..., M_L'\}$ and M_l' is the number of remaining filters or neurons in l_th layer. To get the optimal set, there are two problems required to be solved: (1) Determine the number of neurons or filters of each layer to prune; (2) How to prune redundant neurons or filters.

For global pruning, the objective function is easier and can be defined as follows:

$$\min_{W \in \mathbb{R}^d} F(W^g) = \sum_{n=1}^{N} \frac{D_n}{D} F_n(W_n') \tag{5.3}$$

where W_n' denotes the local model parameters after pruning.

As mentioned in previous subsection, the server aggregates parts of more important neurons or filters from the set of filters or neurons $\{w_l'^1, w_l'^2, ..., w_l'^M\}$ of local models W_n's l_th layer, where $w_l'^m$ represents the m_th neuron or filter in the l_th layer. Therefore, the server is required to find the optimal neurons or filters set for each layer of the global model.

5.2.3 Pruning strategy

Local pruning

As an important model compression method, structured pruning aims to remove the entire filter according to compression requirements and maximize the accuracy of the retained network. Let H_l^m and J_l^m denote the height and width of the feature map of the m_th of the l_th layer respectively. Let $X_{l,m}^{(a)}$ denote the m_th output matrix of the l_th layer for the a_th output feature map, we expand it by row and form it as column map vector as follows:

$$X_{l,m}^{(a)} = \left(x_{l,m,1}^{(a)}, x_{l,m,2}^{(a)}, ..., x_{l,m,s_l}^{(a)}\right) \tag{5.4}$$

in which $S_l^m = H_l^m \times J_l^m$ and the a_th output feature map contains the specific feature information identified by the a_th filter. The size of $X_{l,m}^{(a)}$ depends on feature map size and number of channels. If the l_th layer is a convolutional layer with 32 filters, the size of the feature map output by each filter is 28×28, the size of $X_{l,m}^{(a)}$ is $32 \times 28 \times 28$. To better fit the distributed setting, batch normalization accelerates the network training by reducing internal covariate shift and the L_2-norm keeps the weights of the neural network from deviating too much, thereby improving the consistency and generalization ability of the overall model. This is important because different models may learn different feature representations. The normalization process is as follows:

$$\hat{x}_{l,m,s_l}^{(a)} = \frac{x_{l,m,s_l}^{(a)} - E[X_{l,m}^{(a)}]}{Var[X_{l,m}^{(a)}]}. \tag{5.5}$$

To realize the feature smooth, we add L_2-norm as follows:

$$f(\hat{w}) = \frac{1}{2}\sum_{m=1}^{M}(\sigma(x_{l,m,s_l}^{(a)}) - y^{(a)})^2 + \lambda\sum_{z=1}^{Z}{w_z}^2. \tag{5.6}$$

where w_z represents a weight in the model parameters and Z represents the total number of all weights in the model. These weights form the model's parameter set, which is adjusted through the optimization algorithm during training.

Next, we utilize mutual information (MI) to quantitatively evaluate the mutual influence and importance of each neuron or filter between different layers in the network. To improve the effect and efficiency of pruning, we calculate mutual information layer from the last layer to the first layer of the network since it can directly evaluate the contribution of each filter to the final output. When calculating the MI between each neuron or filter and the final output, we need to consider the space complexity of storing this information. For filters, since they usually exist in the form of multi-channels in convolutional layers, computing and storing the MI of each filter may involve higher space complexity, especially when processing a large number of input channels and output feature maps in deep networks. In contrast, neurons in fully connected layers usually process features that have been abstracted and compressed, so the spatial complexity of MI data associated with these neurons may be relatively low. For a_th output of m_th filter or neuron, the mutual information between it and the final output can be obtained as follows:

$$MI_{l,m} = \sum_{x \in X_{l,m}^{(a)}} \sum_{\hat{y} \in \hat{y}^{(i)}} P(x, \hat{y}) \log \left(\frac{P(x, \hat{y})}{P(x)P(\hat{y})} \right) \tag{5.7}$$

where $P(x)P(\hat{y})$ is the joint probability of $X_{l,m}^{(a)}$ and $\hat{y}^{(i)}$, $P(x)$ and $P(\hat{y})$ are the marginal probability of $X_{l,m}^{(a)}$ and $\hat{y}$ taking a specific value respectively. The kernel density estimation (KDE) method Olsen *et al.* (2024) is used to estimate the probability density function $P(x)$ of continuous variables.

Considering the differences in functions and information across different layers, we calculate the weight of each feature map in a single layer to evaluate its corresponding feature extraction capabilities. This method not only considers the information content of a single feature map but also its relative importance within the layer. Therefore, the mutual information weight of m_th feature map of l_th layer can be described as :

$$MI'_{l,m} = \frac{MI_{l,m}}{\sum_{m=1}^{M} MI_{l,m}}. \tag{5.8}$$

Due to the contingency that a single input cannot reflect the full capability of feature extraction of filters or neurons, and to overcome the randomness caused by a single input sample, we

calculate the average weight of the m_th feature map with O random samples as follows:

$$MI''_{l,m} = \frac{\sum_{o=1}^{O} MI'^{(o)}_{l,m}}{O}. \tag{5.9}$$

We determine the most influential neurons or filters in a neural network by calculating the mutual information between the output of each neuron and the network's final output. Due to the varying significance of weights in different layers, setting a uniform pruning threshold across all layers is challenging. Similar to the method described in Li *et al.* (2019), we arrange the average mutual information $MI''_{l,m}$ of each feature map within a layer in ascending order. We then accumulate these values from the smallest upwards. When the cumulative total surpasses a predefined pruning rate α, we set the threshold for that layer at the last included weight. If a filter's mutual information value is lower than the threshold, it will be removed. This process results in a sequence of thresholds, one for each layer. Neurons or filters falling below these thresholds are pruned, leading to a streamlined and efficient network. This technique not only uncovers the information flow patterns among neurons or filters but also offers a systematic, data-driven strategy for identifying essential versus expendable network components. The local pruning method is given in Algorithm 3.

Global Pruning

The aim of global pruning is to select those filters or neurons with high value of mutual information from among many local models to aggregate a compact global model. Different from traditional FL, clients not only upload their pruned models but also upload the mutual information of each filter or neuron. Compared with model parameters, the amount of mutual information is much smaller, which significantly reduces the communication cost required Estévez *et al.* (2009); Normandin (1991).

We set a maximum pruning rate β_{max} and a minimum pruning rate β_{min}. The maximum pruning rate is set to ensure that the model retains enough capacity to maintain high accuracy. This rate is determined through extensive testing, ensuring that even with significant pruning, the model remains robust without losing crucial features, thus preserving performance. On

Algorithm 3 Structured Pruning Algorithm on clients.

Input: Number of layers L, Number of filters per layer $[M_1, M_2, \ldots, M_L]$, Pruning rate α, Number of random samples O and model W^g
Output: Pruned network model W'.
Initialize:
Initialize $num\_layers \leftarrow L$
Initialize $num\_filters \leftarrow [M_1, M_2, \ldots, M_L]$
Initialize $pruning\_rate \leftarrow \alpha$
Initialize $num\_samples \leftarrow O$
Procedure:
for $l \leftarrow num\_layers - 1$ **to** 0 **do**
  Initialize $MI\_values \leftarrow []$
  for $m \leftarrow 0$ **to** $num\_filters[l]$ **do**
    Initialize $MI\_sum \leftarrow 0$
    for $o \leftarrow 0$ **to** $num\_samples$ **do**
      $X \leftarrow$ ***GetOutputMatrix***(l, m, o)
      $Y \leftarrow$ ***GetFinalOutput***(o)
      $MI\_sum \leftarrow MI\_sum +$ ***CalculateMI***(X, Y)
    end for
    $average\_MI \leftarrow \frac{MI\_sum}{num\_samples}$
    Append $average\_MI$ to MI_values
  end for
  Sort MI_values in ascending order
  Calculate threshold based on $pruning\_rate$
  for $m \leftarrow 0$ **to** $num\_filters[l]$ **do**
    if $MI\_values[m]$ is below threshold **then**
      PruneFilter(l, m)
    end if
  end for
end for

the other hand, the minimum pruning rate is chosen to guarantee sufficient initial pruning, significantly reducing communication overhead and computational costs. This lower bound ensures substantial pruning at the start, essential for achieving noticeable reductions in model size and complexity, which is critical in FL scenarios to enhance communication efficiency without compromising the model's foundational structure and performance. The actual pruning rate of the lg_th layer is $\beta_{lg} \in [\beta_{min}, \beta_{max}]$, and we define the set of pruning rates $Z_{lg} = \{\beta_{min}, \ldots, \beta_{min} + a \times t, \ldots, \beta_{max}\}$, where t and a denotes the t_th communication round and the equal interval respectively and $\beta_{min} + a \times t \leq \beta_{max}$. $M_{lg}^{\prime n}$ denotes the number of remaining filters or neurons of the lg_th layer of local model w_n'. This adaptive strategy

ensures that the network structure is preserved initially and that redundant filters are safely pruned as training progresses. This method balances the trade-off between maintaining high model performance and achieving efficient pruning, allowing the model to adaptively manage complexity and communication volume.

After local pruning, the server receives N local model parameters $W'n$ along with their mutual information sets $MI''1,1, MI''1,2, \ldots, MI''l,m$. The server calculates the average mutual information $MI_l''^{avg}$ of each layer as follows:

$$MI_l''^{avg} = \frac{\sum_{m=1}^{M} MI''_{l,m}}{M}. \tag{5.10}$$

The server then sorts the filters or neurons in each layer in descending order according to their mutual information values. Based on the current pruning rate, a certain proportion of neurons with the highest mutual information is selected to remain in the model. In successive communication rounds, the server monitors changes in the mutual information of each layer. If the current mutual information of a certain layer is not lower than that of the previous round, it indicates that pruning this layer has not significantly impacted network performance. In this case, the server will appropriately increase the pruning rate of this layer to further reduce model complexity while maintaining the network's information capture capability. The global pruning method is shown in Algorithm 4.

5.2.4 Coordination Mechanism

To ensure that client-side and server-side pruning work together effectively, a coordination mechanism is implemented. After client-side pruning, mutual information data is shared with the server, allowing the server to align its pruning strategy with the decisions made by the clients. The server then dynamically adjusts its pruning based on the global model's performance, ensuring that improvements in communication efficiency do not compromise model accuracy. A feedback loop further ensures that the pruned global model sent back to clients aligns with local models, maintaining coherence and effectiveness across the pruning stages.

Algorithm 4 Global Pruning Method

Input: Set of local model parameters $\{W'_1, W'_2, \ldots, W'_N\}$, mutual information sets $\{MI''_{1,1}, MI''_{1,2}, \ldots, MI''_{l,m}\}$, maximum pruning rate β_{max}, minimum pruning rate β_{min} and Communication round T.

Output: Pruned global model W^g (The client will train the whole pruned global model W^g in next round.)

Initialize global model W^g
for Communication round $t = 1$ to T **do**
  Local training for E epoch
  Algorithm 1 for local pruning
  Receive model parameters and mutual information value sets.
  for each layer $l = 1$ to L **do**
    Calculate average mutual information with Eq. 5.10
    Sort filters or neurons in layer l in descending order of $MI''_{l,m}$
    Determine the set of pruning rates $Z_{lg} = \{\beta_{min}, \ldots, \beta_{max}\}$
    for each β_{lg} in Z_{lg} **do**
      Select the top $(1 - \beta_{lg}) \times 100\%$ of filters or neurons to keep in W^g
      if current layer's $MI_l^{''avg}$ is not lower than the previous round **then**
        $\beta_{lg} = \beta_{lg} + t \times a$
      end if
    end for
  end for
  $W^g(t+1) \leftarrow \sum_{n=1}^{N} \frac{D^n}{D} W_n(t)$
  Return $W^g(t+1)$ to client
end for

Through this mutual information-based pruning strategy, we achieve an optimal balance between reducing model complexity and maintaining network performance. The dynamic nature of this strategy is reflected in the gradual adjustment of the pruning rate, allowing the model to incrementally achieve structural simplicity without sacrificing its core information processing capabilities. This method not only improves the computational efficiency of the model but also retains features that are critical to final performance.

5.3 Experiments and results

5.3.1 Data sets

To demonstrate the effectiveness of our pruning method, we compare it with state-of-the-art methods such as FedAvg McMahan *et al.* (2017), FAFED Wu *et al.* (2023c), Fedlab Zeng *et al.* (2023), Flado Liao *et al.* (2023), and pruning methods in FL such as PruneFL Jiang *et al.* (2022a), FedTiny Huang *et al.* (2023), and FedMP Jiang *et al.* (2022b). The datasets used for comparison include image datasets MNIST Deng (2012), CIFAR100 Xu *et al.* (2015), FMNIST Xiao *et al.* (2017), CelebA Liu *et al.* (2015), and traffic-related datasets GTSRB (German Traffic Sign Recognition Benchmark) Stallkamp *et al.* (2011), KITTI Vision Benchmark Suite Geiger *et al.* (2012), and BDD100K (Berkeley DeepDrive Dataset) Yu *et al.* (2020). These traffic datasets are specifically relevant to ITS as they involve tasks such as traffic sign recognition, pedestrian detection, and various driving scenarios. We employed Convolutional Neural Networks (CNN) for the MNIST and FMNIST datasets, VGG9 for the CelebA dataset, and ResNet18 for the CIFAR100, GTSRB, KITTI Vision Benchmark Suite, and BDD100K datasets. For KITTI, we derived a classification task from the object detection annotations. Specifically, we selected seven major traffic participant categories as labels: Car, Van, Truck, Pedestrian, Person_sitting, Cyclist, and Tram. This formulation transforms KITTI into a seven-class recognition problem that reflects diverse traffic participants in real-world road scenes, while discarding ambiguous categories such as Misc and DontCare. For BDD100K, we constructed a weather classification task using the environment annotations provided in the dataset. Each image is labeled with one of several weather conditions, among which we selected sunny, rainy, snowy, foggy and overcast as classification labels.

5.3.2 Setting

The hyper-parameters setting and the model architectures are shown in Table 5.1. Table 5.2 shows the structure of the neural network models used in the clients. The value α of Dirichlet distribution Minka (2000) was used to realize different levels of non-IID. All test experiments

TABLE 5.1. Setting of hyper-parameters and data sets

	MNIST-CNN	FMNIST-CNN
Number of labels	10	10
Number of training samples	5500	60000
Number of testing samples	1000	10000
Learning rate η	0.02	0.02
Local batch_size B	32	32
Communication rounds T	100	100
Local epoch E	10	10
	CIFAR100-ResNet18	Celeba-VGG9
Number of labels	10	40
Number of training samples	50000	141819
Number of testing samples	10000	60780
Learning rate η	0.01	0.01
Local batch_size B	128	64
Communication rounds T	400	200
Local epoch E	20	20

TABLE 5.2. Architecture of different local models.

Architecture	Convolutional layer	Full connected layer
CNN	32, pool 64, pool	2048,10 (input: 2048)
VGG9	$32, 64$, pool 2×128, pool 2×256, pool	$256, 256, 10$ (input:256)
ResNet18	64, pool $2 \times [64, 64]$, $2 \times [128, 128]$, $2 \times [256, 256]$, $2 \times [512, 512]$	avgpool, 100 (input:512)

were run five times for a fair result. The pruning rate of our method on the client or server is set to 40% for comprehensive comparison with other methods. All experiments were on a server with NVIDIA GeForce RTX 3060, Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz and 12GB of VRAM. To mimic the realistic FL environment, we performed aggregation and pruning operations on the CPU while training the models on the GPU.

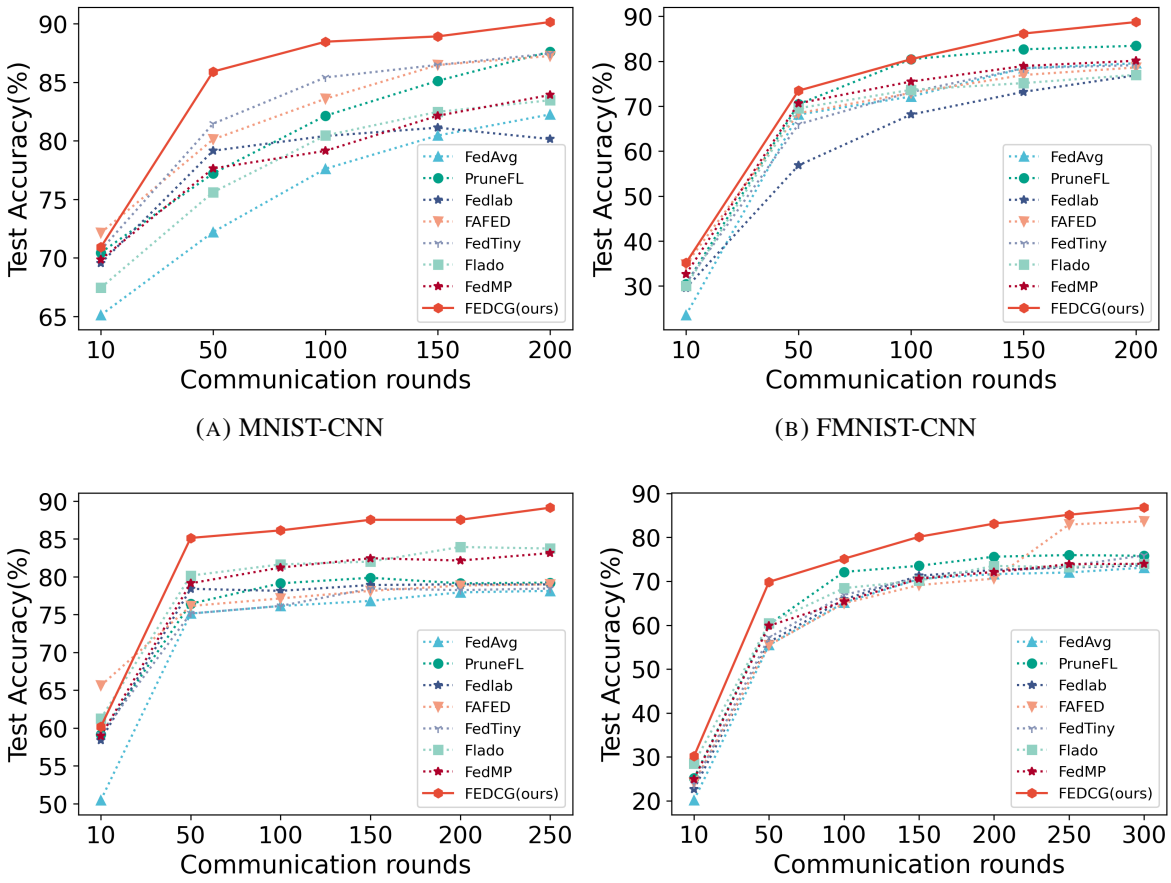
5.3.3 Test accuracy

Figure 5.3 illustrates the test accuracy of various FL methods across different data sets where the value a of Dirichlet distribution is set to 0.2 following many references Li *et al.* (2023); Tan *et al.* (2023); Wu *et al.* (2023a). Remarkably, FEDCG consistently attains the highest test accuracy across all datasets, thereby showcasing its formidable performance across diverse tasks and datasets. On the MNIST, FMINIST, CIFAR100, and Cleba data sets, our approach surpasses competing methods, achieving impressive accuracy levels while requiring fewer communication rounds. Compared with FedAvg (without pruning), FEDCG removes unimportant parameters to generalize better on unseen data. In contrast to conventional techniques, FEDCG not only prioritizes local optima but also meticulously considers the significance of each parameter in shaping the global model. This personalized parameter selection strategy significantly contributes to optimizing each client's impact on the global model, ultimately enhancing overall performance.

5.3.4 Communication costs

Communication overhead stands as critical metrics in elucidating the intricacies of communication costs within the realm of FL. In our endeavor to underscore the commendable attributes of our approach, we have diligently conducted a meticulous examination of parameter counts across a spectrum of communication rounds. As Figure 5.4 aptly portrays, our method, FEDCG, consistently demonstrates exceptional proficiency in curtailing communication overhead across all scrutinized data sets. This significant reduction in the volume of communicated data during each interaction assumes paramount importance, playing an instrumental role in alleviating the overall burden of communication expenses and concurrently elevating the operational efficiency of the FL paradigm. This laudable achievement owes its efficacy to the strategic adoption of our dual-pruning approach.

This dual-pruning strategy, a hallmark feature of our methodology, encompasses two pivotal facets. Firstly, it entails the implementation of localized model pruning at each client's end, thereby effectively attenuating the data payload disseminated during the upload phase.



(A) MNIST-CNN (B) FMNIST-CNN

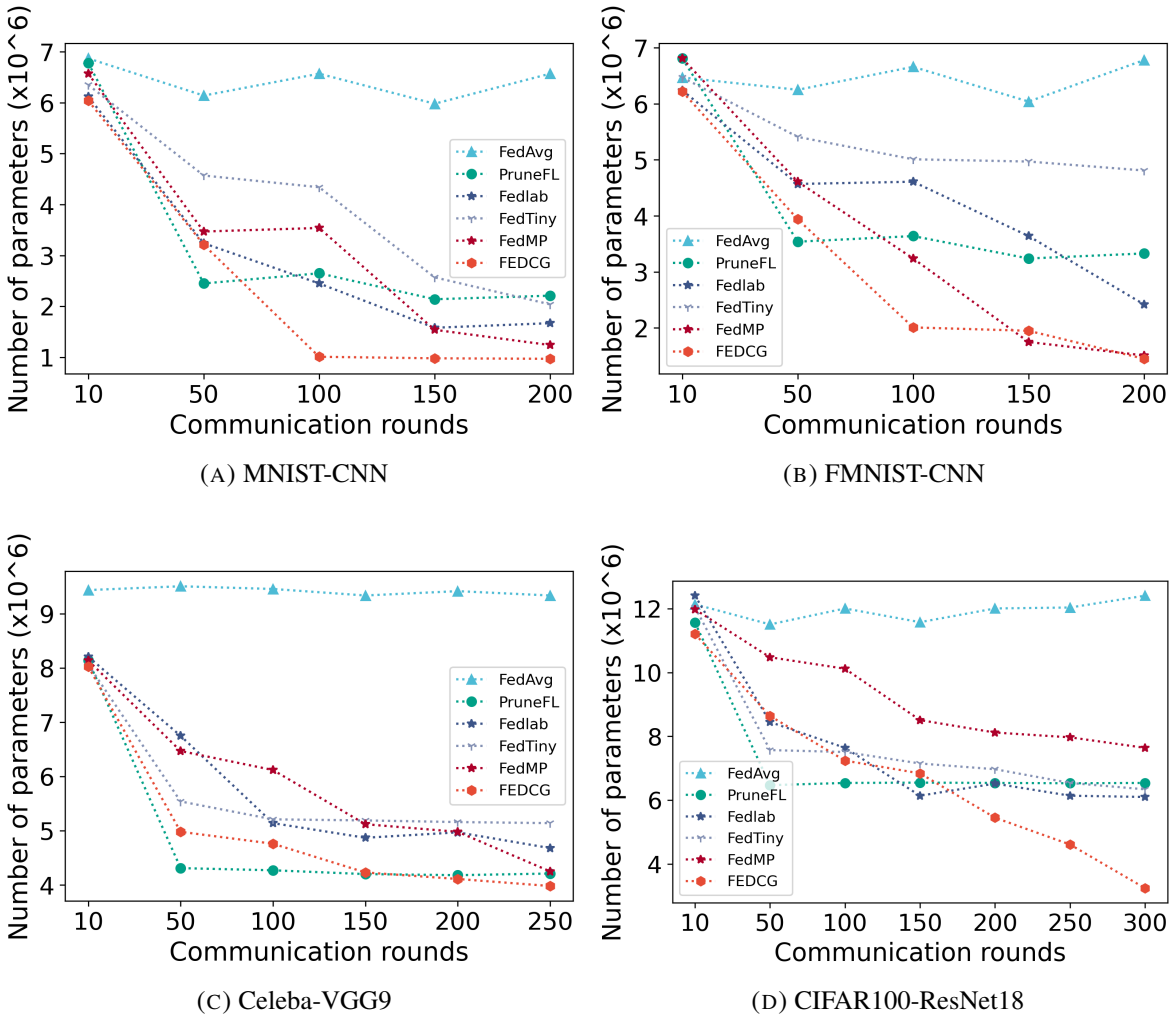
(C) Celeba-VGG9 (D) CIFAR100-ResNet18

FIGURE 5.3. Test accuracy on different data sets. Our proposed **FEDCG** method wins the highest accuracy across four data sets.

Secondly, on the server side, model updates originating from individual clients undergo meticulous pruning, ultimately resulting in the refinement of the global model. This judicious pruning approach engenders a profound reduction in both the quantity of downloaded data and the volume of communicated information, thereby manifesting itself in the striking reductions witnessed in communication overhead, as empirically exemplified.

5.3.5 Robustness under different degrees of Non-IID

Robustness stands as a crucial criterion when assessing the effectiveness of FL algorithms, primarily due to the inherent challenge posed by data heterogeneity. Our investigation revealed



(A) MNIST-CNN (B) FMNIST-CNN

(C) Celeba-VGG9 (D) CIFAR100-ResNet18

FIGURE 5.4. Communication overhead on different data sets. Our proposed **FEDCG** method requires the least parameters for training under data sets with different neural networks.

a general decline in test accuracy across all algorithms when exposed to different datasets and various degrees of non-independent and identical distribution (non-IID), as depicted in Table 5.3. To provide a more quantitative description of this performance degradation, we employed a baseline approach, considering a non-IID degree of 0 (corresponding to IID data). We then calculated the relative changes in accuracy for different non-IID degrees.

Taking the GTSRB dataset as an example, when the non-IID degree is set to 0.5, FEDCG exhibits a 0.45 percentage point decrease in accuracy relative to IID data. In stark contrast,

TABLE 5.3. Robustness of different methods under different degrees of Non-IID. The maximum values are indicated in bold. The values in parentheses indicate the decrease in test accuracy compared to the IID scenario under different non-IID conditions.

MNIST (%)							
	Fedavg	PruneFL	Fedlab	FedTiny	Flado	FedMP	FEDCG
IID	87.15	90.45	89.41	90.47	90.75	90.14	**92.14**
a=0.5	85.41 (-1.74)	90.41 (-0.04)	85.48 (-3.93)	88.74 (-1.73)	85.68 (-5.07)	85.41 (-4.73)	**91.48 (-0.66)**
a=0.2	82.25 (-4.90)	87.61 (-2.84)	80.15 (-9.26)	87.45 (-3.02)	83.46 (-7.29)	83.91 (-6.23)	**90.14 (-2.00)**
a=0.1	80.71 (-6.44)	81.15 (-9.30)	79.54 (-9.87)	85.41 (-4.06)	82.84 (-7.91)	80.78 (-9.36)	**90.05 (-2.09)**
FMNIST (%)							
	Fedavg	PruneFL	Fedlab	FedTiny	Flado	FedMP	FEDCG
IID	85.14	88.57	88.13	89.74	89.14	87.69	**90.15**
a=0.5	81.68 (-3.46)	85.97 (-2.60)	82.67 (-5.46)	84.57 (-5.17)	82.76 (-6.38)	83.48 (-4.21)	**89.64 (-0.51)**
a=0.2	79.54 (-5.60)	83.44 (-5.13)	76.87 (-11.26)	79.14 (-10.60)	76.98 (-12.16)	80.11 (-7.58)	**88.72 (-0.57)**
a=0.1	75.68 (-9.46)	80.74 (-7.83)	75.46 (-12.67)	74.56 (-15.18)	71.56 (-17.58)	79.45 (-8.24)	**88.17 (-1.98)**
CIFAR100 (%)							
	Fedavg	PruneFL	Fedlab	FedTiny	Flado	FedMP	FEDCG
IID	82.41	80.45	84.41	86.87	86.45	86.97	**87.57**
a=0.5	79.41 (-3.00)	78.48 (-1.97)	76.45 (-7.96)	78.45 (-8.42)	80.54 (-6.91)	80.64 (-6.33)	**87.01 (-0.56)**
a=0.2	73.11 (-9.30)	75.81 (-4.64)	73.64 (-10.77)	75.94 (-10.93)	73.99 (-12.46)	73.99 (-13.98)	**86.81 (-0.76)**
a=0.1	70.41 (-11.00)	72.45 (-8.12)	70.48 (-13.93)	72.41 (-14.46)	72.51 (-13.94)	70.15 (-16.82)	**86.41 (-1.16)**
Cleba (%)							
	Fedavg	PruneFL	Fedlab	FedTiny	Flado	FedMP	FEDCG
IID	89.45	90.41	89.71	89.74	90.11	90.45	**91.45**
a=0.5	80.64 (-8.81)	85.41 (-4.00)	82.64 (-6.07)	86.54 (-3.20)	88.15 (-2.96)	84.76 (-5.69)	**90.57 (-0.88)**
a=0.2	78.14 (-11.31)	79.22 (-10.19)	78.99 (-10.72)	78.45 (-11.29	83.74 (-6.37)	83.14 (-7.31)	**89.14 (-0.31)**
a=0.1	75.64 (-13.81)	78.45 (-9.96)	76.46 (-13.25)	76.45 (-13.29)	80.45 (-9.66)	80.46 (-9.99)	**89.05 (-0.40)**
GTSRB (%)							
	Fedavg	PruneFL	Fedlab	FedTiny	Flado	FedMP	FEDCG
IID	89.14	88.87	89.11	89.45	89.30	88.90	**90.75**
a=0.5	86.50 (-2.64)	87.60 (-1.27)	86.80 (-2.31)	88.00 (-1.45)	87.80 (-1.50)	87.20 (-1.70)	**90.30 (-0.45)**
a=0.2	84.30 (-4.84)	85.80 (-3.07)	84.90 (-4.21)	86.50 (-2.95)	86.30 (-3.00)	85.80 (-3.10)	**89.90 (-0.85)**
a=0.1	82.00 (-7.14)	84.50 (-4.37)	82.70 (-6.41)	85.00 (-4.45)	84.80 (-4.50)	84.20 (-4.70)	**89.40 (-1.35)**
KITTI Vision Benchmark Suite (%)							
	Fedavg	PruneFL	Fedlab	FedTiny	Flado	FedMP	FEDCG
IID	85.85	85.40	85.70	86.00	86.30	85.50	**86.50**
a=0.5	82.50 (-3.35)	84.40 (-1.00)	83.80 (-1.90)	84.70 (-1.30)	84.00 (-2.30)	84.50 (-1.00)	**86.00 (-0.50)**
a=0.2	80.00 (-5.85)	81.60 (-3.80)	80.50 (-5.20)	81.90 (-4.10)	82.00 (-4.30)	81.50 (-4.00)	**85.50 (-1.00)**
a=0.1	77.50 (-8.35)	79.30 (-6.10)	78.50 (-7.20)	79.40 (-6.60)	79.50 (-6.80)	79.00 (-6.50)	**85.00 (-1.50)**
BDD100K (%)							
	Fedavg	PruneFL	Fedlab	FedTiny	Flado	FedMP	FEDCG
IID	85.00	84.50	84.70	85.20	85.50	84.80	**86.00**
a=0.5	82.00 (-3.00)	83.40 (-1.10)	82.80 (-1.90)	83.70 (-1.50)	83.00 (-2.50)	83.50 (-1.30)	**85.50 (-0.50)**
a=0.2	80.00 (-5.00)	81.60 (-2.90)	80.70 (-4.00)	81.90 (-3.30)	82.00 (-3.50)	81.50 (-3.30)	**85.00 (-1.00)**
a=0.1	77.50 (-7.50)	79.30 (-5.20)	78.50 (-6.20)	79.40 (-5.80)	79.50 (-6.00)	79.00 (-5.80)	**84.50 (-1.50)**

other algorithms experience significantly greater declines, with the maximum reaching 2.64 percentage points. Similar trends emerge at non-IID degrees of 0.2 and 0.1, with FEDCG maintaining relatively modest decreases of 0.85% and 1.35%, respectively, while other algorithms suffer more substantial drops, ranging from 3.07% to 7.14%. The same observations hold true across the KITTI Vision Benchmark Suite and BDD100K datasets. For instance, on the KITTI dataset, FEDCG shows only a 0.50% decrease at a non-IID degree of 0.5, compared

to a maximum drop of 3.35% for other algorithms. At a non-IID degree of 0.2, FEDCG's accuracy decreases by just 1.00%, whereas other methods experience declines up to 5.85%.

Regardless of the dataset in question, FEDCG consistently demonstrates superior robustness by exhibiting smaller accuracy degradations than its counterparts under different degrees of non-independence and identical distribution. This underscores the potency of our proposed algorithm in handling data heterogeneity more effectively than other methods. The underlying reason for this advantage lies in our algorithm's meticulous consideration of local model parameters' contributions to the global model during the pruning process. Even when local model pruning occurs, it exerts minimal adverse impact on the overall performance of the global model. This unique parameter selection strategy ensures the maximization of each client's contribution to the global model, thereby enhancing overall performance.

5.3.6 Training time to reach target accuracy

To highlight the computational efficiency of the FEDCG algorithm, we compared the computational resource burden required by different algorithms to achieve a specific accuracy. Through dedicated time and computational complexity experiments, we established the resource and time consumption required by various algorithms to achieve 70% and 80% accuracy when processing different data sets. The results on the MNIST data set (Table 5.4) show that the FEDCG algorithm only takes 4364 seconds and 1.2 TFLOPs to achieve 70% accuracy, and also shows the lowest time and computing resource consumption (8944 s and 5.4 TFLOPs on the 80% accuracy). For the more complex CIFAR100 data set (Table 5.5), the algorithm also shows excellent performance. The time and TFLOPs consumed when reaching 70% and 80% accuracy respectively are better than other comparison algorithms. The FEDCG algorithm uses mutual information to evaluate the actual contribution of parameters to the model. It can identify and prune parameters with low information contribution and reduce invalid calculations performed on the client.

TABLE 5.4. Time and accumulated FLOPs per client to reach target accuracy on MNIST data set.

Methods	Time (FLOPs) to reach 70% accuracy	Time (FLOPs) to reach 80% accuracy
FedAvg	18364 s (3.6 TFLOPs)	57415 s (10.6 TFLOPs)
FAFED	17145 s (3.5 TFLOPs)	49512 s (9.9 TFLOPs)
Fedlab	17944 s (3.5 TFLOPs)	49444 s (9.8 TFLOPs)
Flado	9924 s (2.7 TFLOPs)	31254 s (8.1 TFLOPs)
PruneFL	8664 s (1.6 TFLOPs)	12846 s (6.5 TFLOPs)
FedTiny	7347 s (1.6 TFLOPs)	11761 s (6.5 TFLOPs)
FedMP	9364 s (1.9 TFLOPs)	13455 s (6.8 TFLOPs)
FEDCG	4364 s (1.2 TFLOPs)	8944 s (5.4 TFLOPs)

TABLE 5.5. Time and accumulated FLOPs per client to reach target accuracy on CIFAR100 data set.

Methods	Time (FLOPs) to reach 70% accuracy	Time (FLOPs) to reach 80% accuracy
FedAvg	28674 s (5.8 TFLOPs)	79815 s (17.9 TFLOPs)
FAFED	27459 s (5.6 TFLOPs)	73514 s (16.8 TFLOPs)
Fedlab	28344 s (5.6 TFLOPs)	73444 s (16.7 TFLOPs)
Flado	15924 s (4.2 TFLOPs)	41254 s (12.1 TFLOPs)
PruneFL	13864 s (3.2 TFLOPs)	19846 s (9.7 TFLOPs)
FedTiny	11647 s (3.2 TFLOPs)	17661 s (9.7 TFLOPs)
FedMP	14964 s (3.7 TFLOPs)	20455 s (10.2 TFLOPs)
FEDCG	6694 s (2.5 TFLOPs)	13944 s (8.2 TFLOPs)

5.3.7 Ablation study

To investigate the role and importance of various components in the FEDCG pruning method, we designed a series of ablation experiments. These experiments aim to analyze the impact of client-side pruning, server-side pruning, and their combination on model performance and communication efficiency.

Test accuracy with different pruning rate

We determined an original model (VGG9) without any pruning as a baseline. As shown in Table 5.6, we studied the impact of different client pruning rates and server pruning rates on test accuracy. Pruning only on the client side results in significant performance degradation of

TABLE 5.6. Comparison of the test accuracy of 5 control groups.

	Pruning rate (%)				
Local model	0	0.2	0.4	0.2	0.4
Test accuracy(%)	92.11	91.03	88.14	91.09	88.82
Global model	0	0	0	0.2	0.4
Test accuracy(%)	78.10	75.10	72.20	89.33	88.25

the global model compared to the baseline case without pruning. Key network structures may be removed during the pruning process, which will adversely affect the overall performance of the global model. Only performing client-side pruning mainly ensures that the performance of the local model is maintained in the client domain. In contrast, when pruning is applied on both the client and server sides, our observations indicate that the performance of the local model is not far behind that of the global model. Furthermore, the key finding of little performance degradation compared to the unpruned baseline case highlights the effectiveness of pruning simultaneously on the client and side servers to achieve a subtle trade-off between reducing model size and maintaining performance.

Local accuracy vs Global accuracy

To determine the optimal local pruning rate, we conducted a detailed study on the changes in the parameters and performance of each layer of the neural network model under different pruning rates, as shown in Figure 5.5. As the pruning rate increases, the accuracy of the model gradually decreases, which is expected since more parameters being pruned may lead to information loss. It is worth noting that within a certain pruning rate range, the accuracy decreases relatively slowly, indicating that the model can still maintain good performance under a certain pruning rate. For each convolutional layer of the network (cov1-cov10), as the pruning rate increases, their values show a downward trend. This is because pruning leads to a reduction in the corresponding filters. This series of experimental results provides important reference and guidance for us to find the optimal pruning rate.

The number of clients is key to the scalability of FL algorithms, especially when dealing with heterogeneous data from different sources. To evaluate the scalability and robustness of the FEDCG method, we tested and compared accuracy, communication overhead, and training

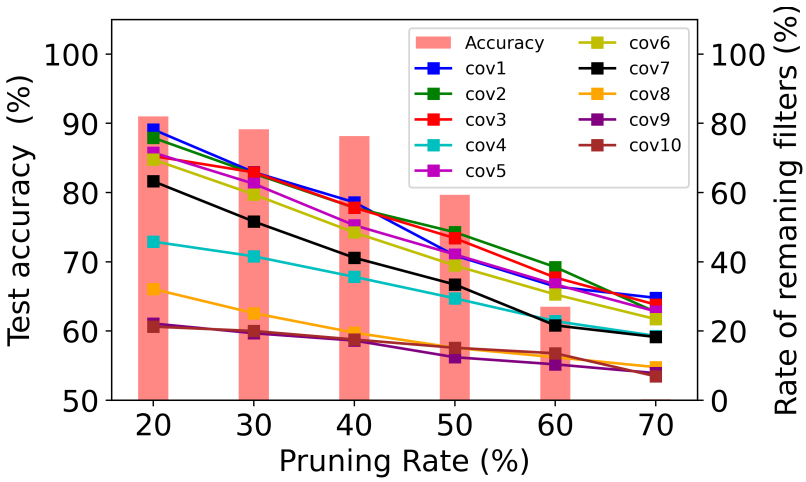


FIGURE 5.5. The accuracy and the number of parameters in each convolution layer of VGG9 and under different pruning rates.

time as the number of clients increased across seven datasets. Table 5.7 shows that FEDCG maintains high accuracy across all datasets despite increasing client numbers. For example, on the MNIST dataset, accuracy only decreased slightly from 91.21% to 90.14% as the number of clients increased from 10 to 100. Similar trends were observed on other datasets, highlighting FEDCG's robustness. Communication overhead and training time naturally increase with more clients. For instance, on the GTSRB dataset, communication overhead rose from 2.10E+01 Mb to 1.68E+02 Mb, and training time from 1.51E+02 s to 1.21E+03 s, from 10 to 100 clients. However, these increases are manageable given the maintained accuracy. The dual pruning strategy based on mutual information helps manage model complexity and communication volume effectively. This ensures that as the number of clients grows, the algorithm dynamically adjusts to optimize the global model without sacrificing performance. In summary, FEDCG demonstrates excellent scalability and robustness, making it suitable for large-scale FL with heterogeneous data sources.

5.4 Conclusion

Our proposed **FEDCG** algorithm achieves a favorable balance between communication efficiency and model performance through a two-stage pruning strategy. In this chapter, we extend the study to simulation environments with a significantly larger number of clients,

TABLE 5.7. The impact of number of clients on the performance of our FEDCG on different data sets. The heterogeneous environment is created by the Dirichlet distribution and the non-IID degree is set as 0.2.

Dataset	Number of clients	Accuracy (%)	Comm. (/Mb)	Training time (/s)
MNIST	10	91.21	2.10E+01	1.51E+02
	20	90.18	4.20E+01	3.02E+02
	50	90.24	8.41E+01	6.04E+02
	100	90.14	1.68E+02	1.21E+03
FMNIST	10	88.54	2.32E+01	2.29E+02
	20	87.64	4.64E+01	4.58E+02
	50	88.21	9.28E+01	9.15E+02
	100	88.72	1.86E+02	1.83E+03
CIFAR100	10	88.14	3.78E+04	4.25E+03
	20	89.41	7.55E+04	8.50E+03
	50	89.31	1.51E+05	1.70E+04
	100	89.14	3.02E+05	3.40E+04
CelebA	10	86.12	2.50E+04	3.03E+04
	20	85.97	5.00E+04	6.05E+04
	50	86.71	1.00E+05	1.21E+05
	100	86.81	2.00E+05	2.42E+05
GTSRB	10	90.00	2.10E+01	1.51E+02
	20	89.50	4.20E+01	3.02E+02
	50	89.10	8.41E+01	6.04E+02
	100	89.00	1.68E+02	1.21E+03
KITTI	10	85.50	2.32E+01	2.29E+02
	20	84.60	4.64E+01	4.58E+02
	50	84.80	9.28E+01	9.15E+02
	100	84.50	1.86E+02	1.83E+03
BDD100K	10	85.00	3.78E+04	4.25E+03
	20	84.50	7.55E+04	8.50E+03
	50	84.30	1.51E+05	1.70E+04
	100	84.00	3.02E+05	3.40E+04

a setting that closely reflects ITS, where thousands of vehicles naturally participate as FL clients. Experimental results across multiple datasets confirm that FEDCG effectively reduces both uplink and downlink communication costs while maintaining accuracy, thereby demonstrating the scalability and adaptability of neural network pruning in FL. These findings not only validate the feasibility of pruning under large-scale client participation but also provide a theoretical foundation for applying pruning-based methods to future real-world ITS deployments.

CHAPTER 6

Exploring Statistical Heterogeneity under Slightly Skewed Labels

---

In the previous chapter, we verified the effectiveness and adaptability of neural network pruning in larger FL systems under more complex communication environments. However, communication efficiency is not the only factor affecting FL performance. In practical deployments, communication heterogeneity often coexists with statistical heterogeneity arising from uneven data distributions across clients. Even when communication overhead is effectively reduced, date heterogeneity may accumulate during repeated model aggregation, gradually slowing global convergence and degrading model generalization. Our experiments show that beyond the extensively studied severe non-IID distributions, even slightly skewed labels can accumulate through repeated aggregation, gradually slowing global convergence and undermining generalization. To address this, we propose a strategy that combines Gaussian mixture clustering with oversampling. Gaussian mixture clustering can handle overlapping data points in model parameters, but insufficient client samples may limit local model training. To overcome this, we introduce a Gaussian mixture-based oversampling method to generate additional samples, enhancing the robustness of FL under slightly skewed label scenarios. Our experiments demonstrate that this method outperforms or matches existing approaches, ensuring more reliable and accurate ITS applications.

6.1 Introduction

Beyond communication challenges, another fundamental difficulty in federated learning (FL) is **data heterogeneity**. In intelligent transportation systems (ITS), this issue is particularly pronounced because traffic data are collected from diverse and heterogeneous sources, such

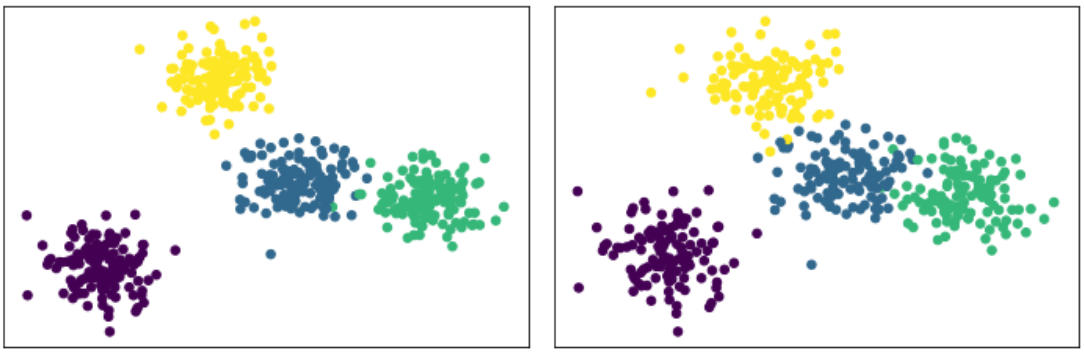
as cameras, sensors, and connected vehicles, which vary in format, sampling frequency, and collection methods. Such diversity inevitably leads to inconsistent data distributions and mismatched feature spaces, thereby increasing the complexity of model training Kim and Mokhtarian (2023). By training models locally and avoiding centralized transfer of raw data, FL can alleviate some of the heterogeneity introduced by multiple data sources Abdelmoniem *et al.* (2023). Indeed, many FL algorithms Heinbaugh *et al.* (2023); Miao *et al.* (2023); Yi *et al.* (2023) have been successfully applied in the transportation domain to address differences between devices and data sources, thereby improving model generalization and robustness.

However, beyond generic data heterogeneity, traffic systems also exhibit another subtle imbalance phenomenon. This phenomenon resembles traditional label skew but differs in degree: rather than certain classes being entirely absent from local datasets, they appear with noticeably imbalanced proportions, leading to *slightly skewed labels*. For example, in traffic flow monitoring, peak-period data constitute a disproportionately large share of the dataset, while off-peak data are relatively scarce. Similarly, in traffic accident detection, minor incidents (e.g., fender-benders) occur much more frequently than severe crashes, even though the latter have a more significant impact. In such cases, all labels remain present, but the distributions are subtly imbalanced. Unlike severe label skew, which often results in extreme dominance by certain classes, this lighter skew is more subtle and harder to detect, yet it still introduces bias during aggregation and reduces the global model's ability to learn from less frequent events.

Slightly skewed labels refer to cases where the label distribution has a slight skew, which, although not extreme, can still impact the training and prediction accuracy of the model. Unlike severe label skew, slightly skewed labels typically manifest as certain labels having slightly fewer or slightly more samples. Although the overall data representativeness remains strong, the model may not focus enough on certain categories during learning, leading to insufficient generalization on those labels. Slightly skewed labels have the following characteristics:

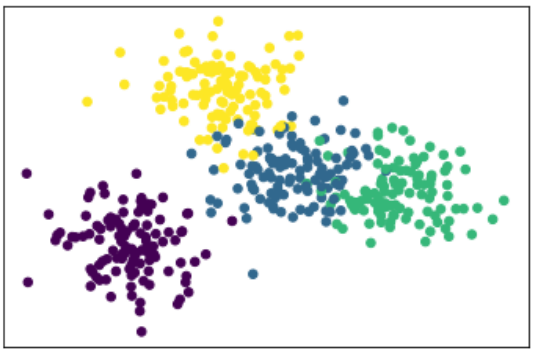
(1) Small differences in label distribution: The data from different devices show subtle differences in label distribution, but these differences are not enough to cause extreme label imbalance.
(2) Potential impact on model training: Although the skew is minor, it may lead to insufficient learning on some less common labels, affecting the model's generalization ability.
(3) Difficult to resolve using traditional methods: Due to the minor degree of skew, traditional resampling or data augmentation methods are often too aggressive in addressing this issue, potentially leading to overfitting or underfitting problems.

Furthermore, in the collected traffic data, the sample number of certain types of events is often significantly smaller than that of other events. This imbalance may stem from multiple factors, such as changes in traffic patterns, differences in the types of events collected by different devices, etc. For example, although minor traffic accidents (such as small rear-end collisions) occur relatively frequently, there may still be insufficient samples of such events in the data set, causing the model to insufficiently learn these categories during training. Data imbalance not only affects the model's ability to learn specific events, but also further exacerbates the problem of slight label drift. Due to the small sample size of some labels, the model may not pay enough attention to these uncommon labels during training, resulting in reduced sensitivity to slight label shift phenomena. In other words, when the model faces slight label drift, the imbalance of the data further complicates this drift problem. To empirically demonstrate the influence of sample size, we adjusted the number of samples of one label on one client and observed an increase in coincident points in Figure 6.1. Labels that are significantly less numerous than others are typically referred to as the **Minority class**. Conversely, the coincident points began to disperse as the number of samples per label increased. Adjusting the sample size on more clients deteriorated the clustering performance further. In contrast to imbalanced samples with the same labels, inadequate samples with skewed labels not only impede local training but also hinder the local model from extracting the features of missing labels, exacerbating the problem.

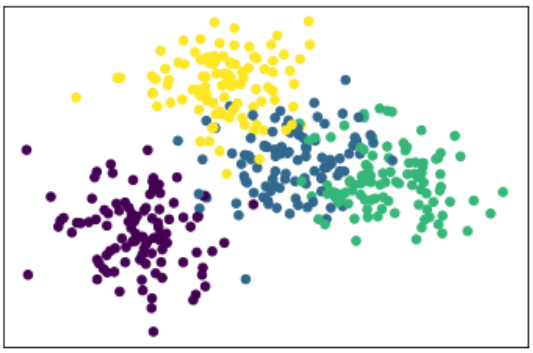


(A) 20%

(B) 40%



(C) 60%



(D) 80%

FIGURE 6.1. The results of local parameters clustering with the number of samples of label "1" decreasing.

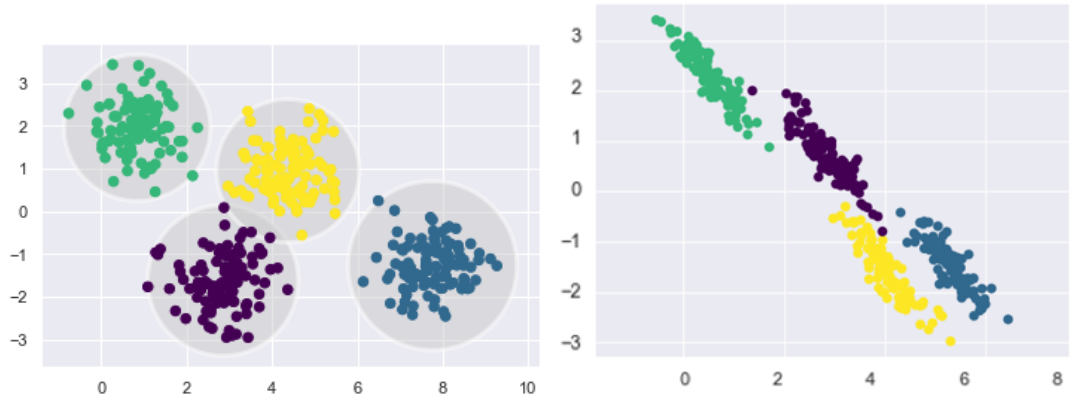
Despite the existence of various algorithms aimed at addressing data imbalance and label skew issues, their effectiveness in handling slightly skewed labels remains insufficient. Traditional resampling and data augmentation methods excel in significant label imbalance but struggle with subtle changes in label distribution. Generative Adversarial Networks (GANs) Li *et al.* (2022b) can generate new samples to balance datasets; however, they may introduce noise and fail to reflect minor label differences accurately. The training process of GANs is also complex and unstable, leading to inconsistencies in sample quality, which affects learning effectiveness. Similarly, diffusion models Lovelace *et al.* (2024) show promise in some applications but struggle with slightly skewed labels due to their generative mechanisms' inability to adjust for minor distribution changes. Knowledge distillation enhances model complexity but does not address subtle label distribution changes. While Clustered Federated Learning (CFL) Chen *et al.* (2020); Ghosh *et al.* (2020); Kumar *et al.* (2020) improves model robustness through local

updates among similar devices, it may overlook nuanced label distributions inherent in slightly skewed labels. For instance, in cases of slightly skewed labels, resampling methods may be too aggressive, resulting in overfitting and failing to capture the subtle dynamics behind slight deviations. As demonstrated in Figure 6.2, our comparison of clustering performance using the MNIST dataset shows that K-means struggles with inter-cluster separation and intra-cluster compactness, especially under minor label drift. In contrast, Gaussian Mixture Clustering calculates the probability of cluster membership, yielding significantly better performance.

These limitations highlight the need for a method that can capture subtle variations in label distributions while remaining robust to overlapping data structures. In this context, the Gaussian Mixture Model (GMM) provides a natural advantage. Unlike hard clustering methods such as K-means, which assign each data point to a single cluster, GMM models the data distribution as a mixture of Gaussian components and estimates the probability that a sample belongs to each cluster. This probabilistic formulation allows GMM to better capture overlapping patterns and subtle distribution shifts, which are typical characteristics of slightly skewed label scenarios. Moreover, GMM can flexibly represent complex data distributions with different covariance structures, making it particularly suitable for modeling heterogeneous client data in federated learning.

However, applying GMM-based methods in FL environments also introduces several challenges. First, client data are distributed across multiple devices, making it difficult to estimate global distribution parameters accurately without violating data privacy constraints. Second, the number of samples available on individual clients may be limited, which can affect the stability of mixture parameter estimation. Third, slightly skewed labels are often subtle and difficult to distinguish from normal statistical variation, making it challenging to identify the appropriate clustering structure. These challenges motivate the design of a federated Gaussian mixture clustering and oversampling framework that can operate under distributed data constraints while preserving the advantages of probabilistic clustering.

In this chapter, we propose a novel approach that combines Gaussian Mixture Clustered Federated Learning (GMCFL) and Gaussian Oversampling (GMO) to effectively address the



(A) K-Means (B) Gaussian mixture

FIGURE 6.2. Two different clustering methods.

challenges of slightly skewed labels and data imbalance in intelligent transportation systems (ITS). The Gaussian Mixture Model (GMM) is employed to identify the underlying data distributions, allowing the model to capture subtle variations in label distributions more accurately. By clustering data based on these distributions, we improve the representation of less common labels, enhancing the model's learning capabilities. Simultaneously, we implement Gaussian Oversampling to increase the number of samples for underrepresented categories, ensuring a more balanced dataset during training. This dual approach mitigates the impact of label skew while improving the overall robustness of the model, leading to better accuracy and generalization in real-world traffic scenarios. Our contributions are summarized as follows:

- We analyzed the influence of slightly skewed labels caused by accidents in ITS. We firstly applied Gaussian mixture techniques to effectively reduce the influence of slightly skewed labels through clustering methods.
- Our Gaussian mixture clustering algorithm overcomes the limitations of k-means when handling overlapping data points caused by high parameter similarity. Additionally, we introduced a Gaussian mixture oversampling algorithm to address data imbalance which generates diverse synthetic samples, improving classifier robustness across different data distributions.

- We theoretically established the convergence of the Gaussian mixture clustering algorithm within the FL environment. Additionally, we demonstrated the stability and convergence of the Gaussian mixture oversampling algorithm, laying a solid theoretical foundation for future research.
- We conducted extensive experiments utilizing data sets from diverse categories, including images, natural language processing, and numerical data. This showcases the versatility and practicality of our methods.

6.2 Gaussian mixture oversampling for generating synthetic samples

To address the challenge of data imbalance, we introduce our proposed Gaussian Mixture Oversampling method in this section. This approach consists of two main steps: find the minority class, and generate the synthetic samples for the minority class.

6.2.1 Find minority class sets

Assume that there are N clients with data $D = D_1, D_2, ..., D_N$ in the system. We let D_n^{maj} and D_n^{min} denote the majority classes and the minority classes of client n, respectively. To find the minority class D_n^{min}, let $p(x|c_n^i)$ donate the probability density function of class c_n^i and the equation is as follows:

$$p(x|c_n^i) = \sum_{j=1}^{M_i} \alpha_{ij} \cdot \mathcal{N}(x|\mu_{ij}, C_{ij}) \tag{6.1}$$

where M_i is the number of Gaussian components used to model class c_n^i, α_{ij} is the weight of the j-th Gaussian component, μ_{ij} is the mean of the j-th Gaussian component, and C_{ij} is the covariance matrix of the j-th Gaussian component. The function $N(x|\mu_{ij}, C_{ij})$ denotes the probability density function of a multivariate normal distribution with mean μ_{ij} and covariance matrix C_{ij}. From Equation (6.1), the density of each class can be obtained:

$$p(c_n^i) = \frac{\sum_{j=1}^{M_i}}{\sum_{l=1}^{L}\sum_{j=1}^{M_l}\alpha_{lj}} \tag{6.2}$$

where L is the number of classes of each client n. This equation computes the probability of the total weights assigned the class c_n^i by summing over the weights of all Gaussian components that belongs to class c_n^i.

To get the global density of class c^i in the FL setting, let $P(x|c^i)$ be global probability density function of the minority class estimated from the local density functions and calculating equation is as follows:

$$P(x|c^i) = \frac{1}{N}\sum_{n=1}^{N} P(x|c_n^i) \tag{6.3}$$

This formula computes the proportion of the total weight assigned to class c_i by summing over the weights of all Gaussian components that belong to class c_i and dividing by the sum of the weights of all Gaussian components in the entire data set. The class with the lowest density is considered the minority class.

6.2.2 Generate the synthetic samples

Each client trains GMM model locally and share the local mean $\mu_{j,n} = \{\mu_{1,n}, \mu_{2,n}, ..., \mu_{M,n}\}$, covariances $C_{j,n} = \{C_{1,n}, C_{2,n}, ..., C_{M,n}\}$ and scalar representing the weight $\alpha_{j,n}$ of the j-th Gaussian components to server. The training process of these parameters will shown in the next section and the central server aggregates those parameters as follows:

$$\begin{aligned}
\mu &= \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{M}\mu_{j,n}, \\
C &= \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{M}C_{j,n}, \\
\alpha &= \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{M}\alpha_{j,n}
\end{aligned} \tag{6.4}$$

When the centre server distributes those parameters to each client, we can generate $\tilde{x}$ data point for client n using the GMM parameters estimated from the local data:

$$\tilde{x} \backsim \mathcal{N}(\mu_{ji,n}, C_{ji,n}), j = 1, 2, .., m. \tag{6.5}$$

where $\mu_{ji,n}$ and $C_{ji,n}$ are the mean and covariance matrix of the j-th Gaussian component estimated from local data of client n. The synthetic samples D'_n are combined into client n. All clients will train on the new data sets and the updated model parameters are sent to the server for aggregation. The more details are shown in **Algorithm 5**.

Algorithm 5 Gaussian mixture oversampling method

Input: Number of class I, Number of Gaussian components M, Number of clients N and Updating rounds T
Output: Synthetic samples sets D'_n.
Client:
for each round $t = 1, 2, 3, ..., T$ **do**
  for each client $n = 1, 2, 3, ..., N$ **do**
    Find the minority class with equation (6.1-6.3)
    Send μ, C and α to server
    Generating the synthetic samples set D'_n (equation (6.5)) from **Aggregation(**μ**,** C**,** α**)**
  end for
end for
Aggregation(μ**,** C**,** α**):**
Update μ, C and α according to equation (6.4)
Return μ, C and α to each client.

6.3 Gaussian mixture clustering for slightly skewed labels

At the round t, the local model of each client can be described as w_t^c. Before training the optimized global model w^g, the best clusters S_k are required to solve as the following

equations:

$$\begin{aligned} &\min F(w^g) = \sum_{1}^{N} \frac{D_n}{D} F(w_n^c) \\ &\max_{\theta^{(k+1)}} Q(\theta|\theta^{(k)}) = \sum_{Z} P(Z|w^c, \theta^{(k)}) \ln P(w^c, Z|\theta) \\ \text{s.t.} \quad &\sum_{1}^{N} \frac{D_n}{D} = 1, \\ &\sum_{1}^{K} \alpha_k = 1, \alpha_k \geq 0, 0 < K \leq N \end{aligned} \tag{6.6}$$

where the second constrain is from Gaussian Mixture model $P(x) = \sum_1^K \alpha_k \centerdot \mathcal{N}(x|\mu_k, C_k)$ and the probability density function is as follows:

$$\phi(x|\mu_k, C_k) = \frac{1}{(2\pi)^{\frac{n}{2}} |C_k|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu_k)^T) C_k^{-1}(x-\mu_k) \tag{6.7}$$

Here, we cluster the local models w^c into different clusters S_k, the latent variable $Z_n \in 1, 2, ..., K$ was introduced to calculate the probability that sample w^c belongs to cluster S_k. The probability distribution laws of Z_k is as follows:

$$P(z_n = k) = \alpha_k \tag{6.8}$$

The function Q has another form:

$$Q(\theta|\theta^{(k)}) = \sum_{n=1}^{N} \left[\sum_{k=1}^{K} P(z_n = k|w_n^c, \theta^{(k)}) \ln P(z_n = k, w_n^c|\theta^{(k)}) \right] \tag{6.9}$$

According to Bayes theory and substitute equation (2) and (4) into equation (4), we obtain:

$$Q(\theta|\theta^{(k)}) = \sum_{n=1}^{N} \left[\sum_{k=1}^{K} P(z_n = k|w_n^c, \theta^{(k)} \ln(\alpha_k \centerdot \phi(w_n^c|\mu_k, C_k)) \right] \tag{6.10}$$

To get the maximum $\theta^{(k+1)}$ of function Q, we let ξ_{nk} donates $\sum_{k=1}^{K} P(z_n = k|w_n^c, \theta^{(k)})$ and take the derivative of μ, C, α as follows:

$$\begin{aligned} \mu_k^* &= \frac{\sum_{n=1}^{N} \xi_{nk}}{\sum_{n=1}^{N} \xi_{nk}}, \\ C_k^* &= \frac{\sum_{n=1}^{N} (w_n^c - \mu_k)(w_n^c - \mu_k)^T \xi_{nk}}{\sum_{n=1}^{N} \xi_{nk}}, \\ \alpha_k^* &= \frac{1}{K} \sum_{n=1}^{N} \xi_{nk} \end{aligned} \tag{6.11}$$

Figure 6.3 provides an overview of the proposed Gaussian Mixture Clustered Federated Learning (GMCFL) method. Due to slightly skewed labels, the parameters of local models are highly similar, leading to the appearance of numerous coincident points in Euclidean space. This situation hinders the clustering of model parameters into their respective clusters. Inspired by the Gaussian mixture equation $P(x) = \sum_1^K \alpha_i \cdot N(x|\mu_i, C_i)$, we assign weight α_i to each local model based on the probability of a sample belonging to a particular cluster. Next, we compute the expectation μ_i and covariance C_i to determine the range of each cluster based on probability rather than the distance between samples and the cluster center. After clustering, the model parameters within each cluster are aggregated, and the aggregated model parameters are further aggregated among clusters. This process is repeated for multiple rounds until the global model converges.

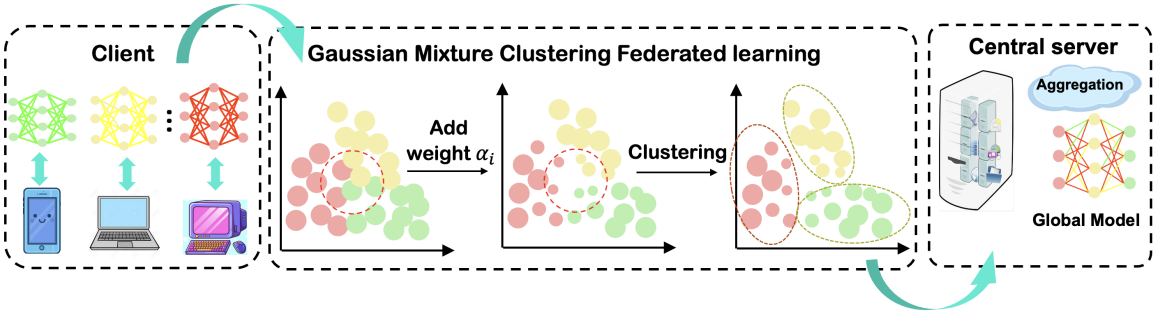


FIGURE 6.3. The workflow of the proposed GMCFL Method.

6.4 Theoretical guarantees

In FL, where data distribution can vary among devices, proving the convergence of the Gaussian mixture sampling algorithm and stability of mixture coefficients, as well as demonstrating the convergence of the Gaussian mixture clustering algorithm, is crucial. These proofs ensure the algorithm can stabilize within a limited number of iterations, yielding reasonable model parameters. Stable mixture coefficients mean the algorithm adapts to different data distributions, enhancing robustness. Proving convergence in the clustering algorithm verifies its performance in diverse data settings. Such proofs guide algorithm and parameter choices for better stability and adaptability in real-world applications. The proof of the following Theorems is provided in the Appendix B.

THEOREM 6.1. ***Convergence of GMO*** *According to Equation (6.6), the approximate objective function obtained by sampling with Gaussian mixture sampling in FL can converge to the expected true objective function,* $F(w) \approx \mathbb{E}_{x\ q}[L(w;x)]$.

THEOREM 6.2. ***Stability of mixture coefficients*** *Due to fluctuations and shifts in the local data distribution, the estimation of the weight and mean for each Gaussian distribution may contain errors, subsequently impacting the performance of the global model. The expected value of* ϵ_n *represents the upper bound of the estimation error in the algorithm, from which we can derive*

$$
\begin{aligned}
(\mathbb{E}[\epsilon_n])^2 \leq \mathbb{E}\Bigg[\max_j \frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\frac{p_n(x_{ni})}{\sum_{j=1}^{J} \hat{p}_j \hat{\alpha}_j(x_{ni})} - \frac{\hat{\alpha}_{j'}}{\sum_{j=1}^{J} \hat{p}_j \hat{\alpha}_j(x_{ni})} (1 - D_{KL}(p_j || p_{j'})) \right)^2 \Bigg] \\
+ \max_{j,j'} \left\{ \mathbb{E}\left[\frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\hat{\alpha}_j \hat{p}_j(x_{ni}) - \alpha_j \frac{p_j(x_{ni})}{p_{j'}(x_{ni})} \right)^2 \right] \right\} \\
+ \mathbb{E}\left[\frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\hat{\alpha}_{j'} \hat{p}_j(x_{ni}) D_{KL}(p_j || p_{j'}) + \frac{\hat{\alpha_{j'}} p_j(x_{ni})}{p_{j'}(x_{ni})} + \frac{\hat{\alpha_{j'}} p_{j'}(x_{ni})}{p_j(x_{ni})} + \alpha_{j'} \right) \right]
\end{aligned}
$$

THEOREM 6.3. ***Convergence of GMCFL*** *In the FL setting, the Gaussian Mixture clustering algorithm converges as*

$$T \geq \frac{1}{2}\left(\frac{L^2}{\epsilon^2} log \frac{2}{\delta} + \frac{log(\frac{2}{\delta})}{\epsilon^2} log\left(\frac{1}{\theta}\sum_{t=1}^{T} exp\left(\frac{\epsilon^2 num}{L^2 T}\cdot t\right)\right)\right) \tag{6.12}$$

when $num \geq \frac{16L^2 log(\frac{2}{\delta})}{\epsilon^2}$. *(The first item is the relationship between the number of samples and the accuracy, and the second item is the theoretical convergence speed.)*

6.5 Experiments

6.5.1 Setting

To evaluate the proposed Gaussian Mixture Model (GMM)-based oversampling approach, we conducted a series of experiments under both slightly skewed label and general non-IID settings. Each experiment was repeated five times with different random seeds to ensure the reliability of the results. Unless otherwise specified, we simulated a federated environment with 100 clients, of which 10% were randomly selected to participate in each training round. The datasets were evenly partitioned among clients to provide a consistent experimental basis.

Slightly skewed label scenario. To emulate the imbalance commonly observed in ITS-like environments, we constructed a slightly skewed label distribution by randomly removing approximately 10% of class samples from each client. This setting preserves all label categories but introduces mild imbalances in class frequencies, enabling us to assess the performance of clustered FL methods under subtle yet realistic data skew.

General non-IID scenario. In addition to slightly skewed labels, we further tested robustness under a standard non-IID configuration generated via a Dirichlet distribution, which induces more severe heterogeneity across clients. This provides a comparative baseline to evaluate the adaptability of our approach under high non-IID conditions.

Implementation details. All experiments were implemented using PyTorch in an actual federated learning system. For inter-process communication, the system employed the

TABLE 6.1. Configuration of sever and clients devices.

	CPU -Intel(R) Core(TM)	GPU-NVIDIA	VARM
Server	i7-12700H CPU @2.30 GHz	RTX 3070	64 GB
Client 1	i7-12700H CPU @2.30 GHz	RTX 3070	32 GB
Client 2	i7-8565U CPU @ 1.80GHz	MX230	8 GB
Client 3	i9-13900HX CPU @ 2.20 GHz	RTX 4060	16 GB
Client 4	i5-13500HX CPU @ 4.70GHz	RTX 4060	32 GB
Client 5	i5-10400F CPU @ 2.90GHz	RTX 3060	12 GB

Message Passing Interface (MPI) Gropp *et al.* (1996). The hardware setup included one tower server and five laptops acting as clients, with detailed specifications summarized in Table 6.1. Local model training was executed on client devices, while aggregation was coordinated by the server, faithfully replicating realistic FL deployments.

6.5.2 Datasets

To comprehensively evaluate the effectiveness and generality of our proposed oversampling method, we conducted experiments across a variety of classification tasks, including computer vision (CV), natural language processing (NLP), and structured datasets from the KEEL repository. These datasets span multiple domains such as industry, healthcare, and language, thereby allowing us to validate the robustness of our approach under diverse application scenarios. For balanced datasets, including image and text classification tasks, we introduced controlled imbalance by randomly removing a portion of class samples, thereby constructing a slightly skewed label distribution to test the resilience of clustered FL methods.

Image classification. We selected FMNIST, CIFAR-10, and CelebA as representative benchmarks of increasing complexity in computer vision. FMNIST consists of 10 fashion categories, CIFAR-10 contains natural images from 10 object classes, and CelebA provides over 200,000 face images annotated with 40 attributes, which we used for attribute classification.

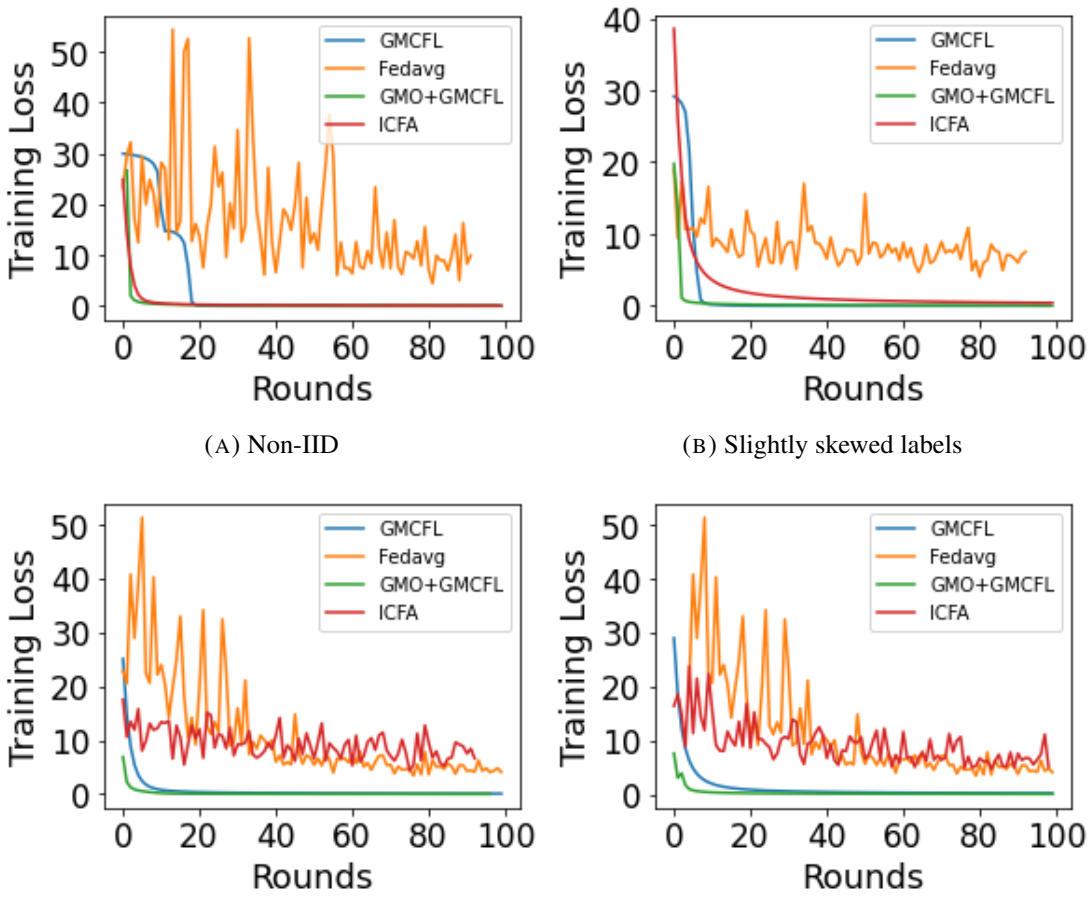
Natural language processing. For NLP tasks, we included datasets from different domains: Books Corpus for text classification, NER (Named Entity Recognition), ADE (Adverse Drug Events), and SSM4H (Social Media Mining for Health Applications). These datasets

cover tasks ranging from general language modeling to domain-specific biomedical and health-related classification.

KEEL datasets. To further validate the method on structured tabular data, we used several datasets from the KEEL repository, including *wisconsin*, *yeast1*, *vehicle*, *shuttle-c0-vs-c4*, *car-vgood*, and *kr-vs-k-one_vs_fifteen*. These datasets are widely used benchmarks for imbalanced classification and provide additional evidence of the versatility of our approach.

6.5.3 Convergence

We presented the results of our GMO and GMO+GMCFL method to validate the theoretical claims in Section 6.4. Training loss directly demonstrated the convergence performance of machine learning algorithm. As shown in Figure 6.4, we test the performance of the proposed method under higher degree non-IID setting and slightly skewed labels with imbalanced data. The convergence curve showed our method not only has the comparable convergence rate with other classic CFL methods under general non-IID setting, but also win the best convergence performance under slight non-IID. When dealing with imbalanced datasets, the unsupervised feature extraction method ICFA does not take class information into account, which may result in features that are biased towards the majority class and ignore the features of the minority class, thereby affecting the performance of the classifier. Additionally, the feature extraction process of ICFA relies on the assumption of independence among the data, which may not hold true when dealing with imbalanced data, leading to poor feature quality. In an imbalanced dataset, due to the small number of samples in the minority class, their probability density function is usually more concentrated and distinct compared to the majority class. This means that Gaussian Mixture Clustering can more accurately cluster the minority class and separate the majority classes with clear boundaries, thus improving the quality of clustering. Furthermore, The GMO method for pre-sampling accelerated the convergence of GMCFL 5% for higher degree Non-IID, 7% for slightly skewed labels and 4% for slightly skewed labels with imbalanced data.



(A) Non-IID

(B) Slightly skewed labels

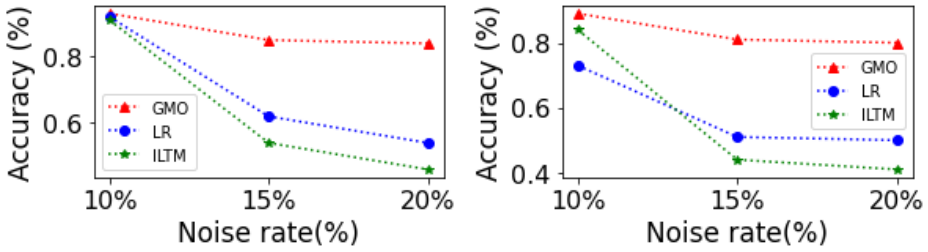
(C) Imbalanced data

(D) Slightly Skewed Labels + Imbalanced data

FIGURE 6.4. The convergence results of different CFL method on MINIST data set under four settings.

6.5.4 Noise Injection

To demonstrate the robustness of the GMO sampling method enhanced GMCFL with noise injection, we test the accuracy with two methodsRoh *et al.* (2021): label flipping and targeted label flipping. Take the MNIST data set as an example, we compared the algorithm performance with various noise rate. Figure 6.5 shown the accuracy of different sampling methods applied in CFL under different noise setting. Our GMO sampling method has the best accuracy compared with others and is the most robust when varying the noise rate.



(A) Label Flipping (B) Targeted label flipping

FIGURE 6.5. The convergence results of different CFL methods on MINIST data set under two noise settings.

6.5.5 Clustering performance

Clustering performance significantly influences federated clustering algorithms, particularly in slightly skewed label scenarios. In this section, we rigorously assess and compare our GMCFL (Gaussian Mixture Clustering) algorithm's performance with other K-Means based algorithms using key metrics: Davies–Bouldin index (DBI), Calinski–Harabasz (CH), and Silhouette Coefficient (SC). The DBI metric measures cluster similarity and separation. Lower DBI values indicate better clustering, implying higher intra-cluster similarity and increased inter-cluster distance. CH assesses the balance between intra-cluster similarity and inter-cluster dissimilarity, with higher values signifying better clustering. The Silhouette coefficient provides an overall measure of clustering quality, where higher average values denote superior clustering. Our results in Table 6.2 consistently highlight the superior performance of GMCFL and GMO+GMCFL across both CIFAR-10 and ADE datasets. This notable achievement stems from our Gaussian mixture clustering approach's unique ability to incorporate data point probability assignments. This precise differentiation of overlapping points enhances clustering accuracy and robustness.

6.5.6 Comparison

To showcase the overall performance of various CFL methods, we conducted a comparison under slightly skewed label settings with different imbalanced ratios (IR), following the methodology outlined in Xie *et al.* (2020). In Table 6.3, we utilized three metrics—Precision,

TABLE 6.2. Comparison results of clustering performance on CIFAR-10 and ADE data sets.

			CIFAR-10			
	ICFA	FAC	FEDKM	MCFL	GMC	GMO+GMC
DBI	1.245	0.972	0.974	0.814	0.645	0.515
CH	5.088	5.124	5.706	5.601	9.647	10.144
SH	0.404	0.367	0.314	0.415	0.914	0.987
			ADE			
	ICFA	FAC	FEDKM	MCFL	GMC	GMO+GMC
DBI	2.145	1.133	1.271	1.101	1.015	0.673
CH	14.403	15.075	10.844	9.745	21.146	27.102
SH	0.419	0.624	0.614	0.965	1.012	1.443

Recall, and F1-score—to assess the performance of these algorithms. Our proposed GMCFL and GMO+GMCFL methods outperform all state-of-the-art algorithms in terms of precision, recall, and F1-score. The equations of these metrics are defined as follows:

$$\begin{aligned} Precision &= \frac{TP}{TP+FP}, \\ Recall &= \frac{TP}{TP+FN}, \\ F1-Score &= 2 \times \frac{Precision \times Recall}{Precision + Recall} \end{aligned} \tag{6.13}$$

For the evaluation, we employed three image classification and NLP data sets, removing 10% of classes on each client to test the performance of diverse algorithms under slightly skewed label settings. Additionally, we randomly selected one label and eliminated 90% of samples to create an imbalanced data distribution. The baseline algorithm, FedAvg, fell short in achieving acceptable precision, as it aggregated local model parameters using average weights. In contrast, other benchmarks demonstrated reasonable precision when compared to the baseline.

When tackling data heterogeneity issues in FL, ICFA Ghosh *et al.* (2020), FedKM Kumar *et al.* (2020), and FAC Goetz *et al.* (2019) algorithms group clients with similar features into several clusters, and train models within each cluster to adapt to non-IID data distributions. MCFL Xiong *et al.* (2024), FFREEDA Shenaj *et al.* (2023), and FedSoft Ruan and Joe-Wong (2022) bolster their robustness across a variety of non-IID scenarios by clustering features

TABLE 6.3. Comparison reults of different clustered FL methods on different machine learning tasks with IID setting.

	FedAvg			ICFA			FAC		
	Precision(%)	Recall(%)	F1-score	Precision(%)	Recall(%)	F1-score	Precision(%)	Recall(%)	F1-score
FMNIST	83.45	76.24	77.96	91.21	71.48	76.54	83.11	69.48	75.69
CIFAR-10	61.25	50.47	55.34	78.25	68.24	71.42	74.42	61.45	67.32
Celeba	72.45	61.45	66.41	84.38	63.52	67.76	84.92	75.46	79.91
BC	70.25	62.15	65.95	82.06	75.64	78.72	82.11	70.58	75.91
NER	69.15	58.64	63.46	71.24	63.47	67.13	81.04	69.78	74.99
ADE	71.15	59.48	64.80	78.10	71.78	74.80	80.14	72.84	76.32
SSM4H	72.45	64.14	68.04	70.82	62.87	66.61	82.28	71.58	76.56
Wisconsin	85.14	76.87	80.79	82.38	69.52	75.41	83.41	74.97	78.97
Yeast1	86.44	75.94	80.85	84.54	71.64	77.56	82.13	71.84	76.64
Vehicle	89.45	80.87	84.94	90.17	85.64	87.85	90.14	82.87	86.35
Shuttle	88.45	79.48	83.73	91.87	80.97	86.08	89.44	81.76	85.43
Car	87.45	77.68	82.28	92.03	87.46	89.69	91.47	85.49	88.38
KR	86.18	72.45	78.72	93.24	86.48	89.73	92.61	80.97	87.26
	FedKM			FFREEDA			MCFL		
	Precision(%)	Recall(%)	F1-score	Precision(%)	Recall(%)	F1-score	Precision(%)	Recall(%)	F1-score
FMNIST	87.64	62.79	65.33	87.68	64.20	74.00	87.71	59.00	70.54
CIFAR-10	73.77	64.21	66.28	72.41	59.17	65.00	76.13	56.45	64.83
Celeba	78.22	64.73	65.74	76.27	63.35	69.00	77.37	58.98	66.94
BC	83.10	62.38	75.63	87.73	57.47	69.00	88.34	56.31	68.78
NER	88.94	55.32	65.16	87.58	62.09	73.00	88.31	58.79	70.59
ADE	85.54	57.66	57.16	83.13	61.74	71.00	85.84	56.75	68.33
SSM4H	87.24	64.72	74.05	84.14	57.82	69.00	86.15	58.19	69.46
Wisconsin	86.97	64.09	71.03	85.09	63.84	73.00	88.24	55.00	67.77
Yeast1	85.35	55.67	64.41	86.68	57.86	69.00	83.61	57.72	68.29
Vehicle	87.82	56.24	53.77	85.17	60.62	71.00	86.60	59.75	70.71
Shuttle	88.90	56.47	57.07	88.99	64.10	75.00	87.48	55.97	68.26
Car	84.78	63.42	63.67	87.69	62.11	73.00	85.95	56.71	68.33
KR	84.83	62.13	75.67	85.62	56.92	68.00	87.29	58.96	70.38
	Fedsoft			GMCFL			GMO+GMCFL		
	Precision(%)	Recall(%)	F1-score	Precision(%)	Recall(%)	F1-score	Precision(%)	Recall(%)	F1-score
FMNIST	90.55	59.40	71.74	90.26	89.93	91.57	94.56	90.58	92.53
CIFAR-10	86.77	58.42	69.83	87.01	85.32	86.15	89.72	90.49	90.10
Celeba	89.96	57.08	69.84	89.43	88.91	90.65	91.92	88.36	90.11
BC	87.57	59.13	70.60	89.08	90.11	89.74	90.42	90.19	90.30
NER	88.44	59.70	71.28	88.75	87.91	90.30	92.23	89.10	90.64
ADE	89.54	59.19	71.27	90.08	90.37	90.22	92.74	90.41	91.56
SSM4H	88.58	58.83	70.70	89.40	93.95	91.62	94.74	94.14	92.83
Wisconsin	86.71	59.36	70.47	88.34	83.14	90.68	93.95	89.00	91.41
Yeast1	89.78	56.67	69.48	88.50	87.90	90.16	92.38	90.95	91.66
Vehicle	87.39	58.50	70.08	88.57	90.27	89.41	90.90	91.63	90.26
Shuttle	87.97	56.65	68.92	89.22	88.83	91.46	94.69	90.06	91.79
Car	90.97	58.81	71.44	90.06	89.84	89.95	90.01	90.89	90.45
KR	86.77	57.32	69.03	90.40	91.48	90.93	94.65	93.26	92.40

Number of clusters is four Sattler *et al.* (2020).

to capture more information about data distributions. However, when dealing with slightly skewed labels, clustering clients becomes a daunting task due to the high similarity in data distributions. The core advantage of our proposed GMCFL method lies in its ability to calculate the probabilities of samples belonging to specific clusters, leveraging Gaussian mixture theory.

Simply using accuracy as a performance metric can cause the model to focus too much on negative examples and ignore positive examples. Therefore, it is more appropriate to use recall as a performance metric. Since our GMCFL algorithm did not cluster clients according to data distribution but focused on the similarity of model parameters, its recall is clearly higher than other methods. The slightly skewed data resulted in poorer clustering performance, while imbalanced data negatively impacted the model's prediction accuracy. Although these data-clustered algorithms demonstrated strong classification performance in terms of precision, their recall values were lower than expected. Data-clustered algorithms are not robust to imbalanced data because the local training of most algorithms minimizes a loss function that is usually computed over the prediction error of the entire training set. Our GMO can generate a new sample with the original characteristics of the sample by using the sample square method, which helps maintain the intrinsic characteristics and structure of the data and provides a high-quality model. It also can generate multiple synthetic samples whose distribution is different from the original data set, thereby increasing the diversity of the data, which is beneficial for improving the robustness and generalization performance of the model.

6.5.7 Ablation study

To further investigate the robustness of the proposed framework, we conduct ablation studies on two key hyperparameters: the number of Gaussian components M and the number of clusters K. The experiments are performed on three representative datasets, including FMNIST, CIFAR-10, and ADE, and the performance is evaluated using Precision, Recall, and F1-score. Table 6.4 presents the results when varying the number of Gaussian components M. When $M = 1$, the Gaussian mixture model degenerates to a single Gaussian distribution, which limits its ability to capture complex local data distributions under slightly skewed label settings, resulting in relatively lower performance. As M increases, the model gains stronger representation capability and the performance improves noticeably. The best results are achieved when $M = 3$, indicating that a moderate number of Gaussian components is sufficient to effectively model heterogeneous client data. When M continues to increase,

TABLE 6.4. Ablation study on the number of Gaussian components M for the proposed GMO+GMCFL framework.

	FMNIST			CIFAR-10			ADE		
M	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
1	86.94	84.52	85.71	82.36	79.84	81.08	86.87	84.91	85.88
2	89.48	87.31	88.38	85.94	83.72	84.82	89.24	87.65	88.44
3	**91.96**	**90.28**	**91.11**	88.73	86.95	87.83	91.15	89.77	90.45
4	91.52	89.84	90.67	**89.84**	**88.21**	**89.02**	**91.93**	**90.62**	**91.27**
5	90.81	89.17	89.98	88.91	87.08	87.99	91.22	89.94	90.58

TABLE 6.5. Ablation study on the number of clusters K in the GMCFL framework.

	FMNIST			CIFAR-10			ADE		
K	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
2	86.32	83.91	85.10	81.74	79.52	80.61	85.94	83.86	84.89
3	89.72	87.41	88.55	85.46	83.51	84.47	88.74	86.95	87.83
4	**92.63**	**90.84**	**91.73**	**89.83**	**88.21**	**89.01**	**92.34**	**91.02**	**91.68**
5	91.74	89.95	90.83	88.96	87.32	88.13	91.52	90.03	90.77
6	90.81	89.12	89.96	88.04	86.41	87.21	90.84	89.36	90.09

the performance becomes relatively stable with only slight fluctuations, suggesting that the proposed method is not overly sensitive to this parameter. Table 6.5 shows the influence of the number of clusters K in the GMCFL framework. When the number of clusters is small (e.g., $K = 2$), clients with different data distributions may be grouped together, which weakens the ability of clustered training to capture statistical heterogeneity. As K increases, the clustering becomes more refined and clients with similar model parameters can be grouped more accurately, leading to improved performance. The best performance is observed when $K = 4$ across all datasets. However, when K becomes larger, the performance slightly decreases because excessive clustering may lead to over-segmentation of clients, reducing the amount of training data available within each cluster and affecting the stability of model aggregation. Overall, the results demonstrate that the proposed method is robust to the choice of hyperparameters, and empirically the configuration $M = 3$ and $K = 4$ provides the best trade-off between model expressiveness and clustering effectiveness.

6.6 Conclusion

This study introduced a FL approach that combines Gaussian mixture clustering with oversampling to address the challenge of slightly skewed labels and data imbalance in ITS. The proposed method enhances model robustness and privacy preservation in non-IID environments, thereby contributing to more reliable traffic management and safety. While limitations remain—for example, potential performance degradation under extreme data imbalance, device disconnections, and the computational complexity of Gaussian mixture clustering in large-scale systems—this work highlights an often-overlooked but practically important aspect of statistical heterogeneity. By explicitly addressing slightly skewed labels, our study complements existing research on severe non-IID distributions and provides a valuable step toward a more comprehensive understanding of data heterogeneity in FL.

CHAPTER 7

System Integration under Compounded Heterogeneity

In the previous chapters, we investigated the effectiveness and adaptability of neural network pruning in improving the efficiency of FL under communication and statistical heterogeneity. However, real-world FL systems rarely face these challenges in isolation. Instead, communication, data, and system heterogeneity often coexist and interact, creating compounded difficulties that significantly affect both convergence and efficiency. This chapter extends our study to such compounded heterogeneity settings, providing a more comprehensive evaluation of pruning strategies. To address this, we propose Federated Segmented Pruning (FedSP), a stage-aware pruning framework that dynamically adjusts pruning intensity throughout training. By coupling pruning schedules with model convergence states, FedSP balances communication efficiency and model accuracy, enabling stable learning under highly heterogeneous conditions. Extensive experiments show that FedSP reduces communication overhead by up to 40% while simultaneously improving global model accuracy by 8% compared to strong FL baselines. These results not only validate the robustness of pruning under compounded heterogeneity but also establish FedSP as a system-level solution for scaling FL to complex and demanding environments.

7.1 Introduction

Real-world distributed environments are far more complex than that we discussed before. Variations in communication bandwidth, computational capacity, and data distribution often coexist and interact, forming a highly coupled system environment. such as ICPS , control

nodes across different production lines or regions may operate under distinct network conditions—some connected via low-bandwidth wireless networks, others via stable industrial Ethernet Ding *et al.* (2025); Jin *et al.* (2025). Embedded devices may also exhibit orders-of-magnitude differences in computation and memory capacity, while the data collected across nodes can vary significantly depending on task type, operating conditions, or sensor modality Rejeb *et al.* (2023). Such multidimensional heterogeneity tightly couples communication latency Jiang *et al.* (2025), computational load, and model convergence dynamics Zhao *et al.* (2025), posing significant challenges for efficient and stable FL in practice. The complexity lies not only in the diversity of these heterogeneities but also in their compounded impact on system-level efficiency.

Under these compounded conditions, FL faces the fundamental challenge of balancing system efficiency. Bandwidth fluctuations lead to delayed parameter uploads and extended synchronization time Du *et al.* (2023); Xia *et al.* (2023) and disparities in computation capability cause highly unbalanced local training durations Martin *et al.* (2025); Tran *et al.* (2019). Additionally, statistical heterogeneity further amplifies global model instability. Existing studies have primarily optimized one aspect in isolation—either communication or computation—while neglecting their interactions Bai *et al.* (2025); Viterbo *et al.* (2025). Fixed-rate pruning methods lack flexibility to adapt to network dynamics Liu *et al.* (2024a); Pervej *et al.* (2024); Salehi *et al.* (2024); Wu *et al.* (2023b); Xiong *et al.* (2023), whereas adaptive pruning schemes rely on frequent bandwidth monitoring and model evaluation Liu *et al.* (2024b); Su *et al.* (2024), which increase computational overhead and undermine stability Liu *et al.* (2024b); Su *et al.* (2024). Therefore, the key challenge is to design a pruning mechanism that can dynamically adapt to both system and model states without requiring real-time feedback.

In earlier chapters, we have already verified the feasibility and adaptability of neural network pruning for improving FL efficiency under different heterogeneous conditions . However, traditional pruning strategies typically adopt fixed or simplistic adjustment rules, overlooking the model's dynamic sensitivity to pruning throughout training. Ignoring this temporal sensitivity leads to mismatches between the pruning schedule and the model's convergence

state: in early stages, aggressive pruning disrupts gradient updates and slows convergence; in later stages, conservative pruning fails to sufficiently compress redundant parameters, wasting both communication and computation resources. Over time, these mismatches significantly degrade global convergence and limit system-level optimization potential.

To address these issues, this chapter introduces a novel framework—Federated Segmented Pruning (FedSP). FedSP exploits the intrinsic variation in model pruning sensitivity across training stages. The training process is divided into three distinct phases, each following a different pruning-rate growth pattern: a linear increase in the early stage to maintain stability, an exponential increase in the middle stage to maximize compression efficiency, and a logarithmic increase in the late stage to preserve accuracy and prevent oscillation. This stage-aware design allows FedSP to implicitly adapt to dynamic network and system conditions without requiring explicit bandwidth feedback, achieving joint optimization of communication and computation under compounded heterogeneity.

From a mechanistic perspective, FedSP improves communication efficiency, computational balance, and model convergence simultaneously. First, the pruning mechanism significantly reduces the number of parameters transmitted per round, mitigating synchronization delays under bandwidth variation. Second, its segmented pruning schedule enables resource-constrained clients to participate using lightweight models, alleviating training latency caused by system heterogeneity. Finally, by maintaining higher model capacity in the early phase and gradually compressing redundant weights in the later phase, FedSP enhances model generalization across non-identically distributed data. By dynamically balancing model complexity, communication overhead, and convergence stability over time, FedSP achieves coordinated optimization across communication, system, and statistical heterogeneity—demonstrating strong adaptability and scalability for complex real-world federated environments.

The three main contributions of this work are as follows:

1. Segmented pruning: We are the first to propose FedSP, a segmented pruning framework for FL that dynamically adjusts pruning rates across different training stages, effectively reducing communication overhead while preserving model performance. It's simple yet

practical design ensures ease of implementation in real-world applications.

2. Theoretical guarantee: We provide a rigorous theoretical analysis, demonstrating that FedSP guarantees convergence and stability under non-IID data distributions and device heterogeneity in ICPS environments.

3. Experimental evaluation: Extensive experiments on diverse datasets validate the effectiveness of FedSP. The results show significant improvements in communication efficiency, model accuracy, and scalability compared to state-of-the-art pruning methods while maintaining a simple and robust design.

7.2 Methods

7.2.1 Overall Framework

We first present a concise framework named Federated Segmented Pruning (FedSP), a stage-aware pruning framework designed for FL in dynamic ICPS environments. FedSP reduces communication overhead without relying on real-time network state information by leveraging the model's varying sensitivity to pruning at different training stages (Figure 7.1). The framework includes three main phases: early pruning, mid-term pruning, and late pruning. These phases are not predefined by fixed communication rounds; instead, their boundaries are determined from the evolution of the global training loss. Specifically, the server smooths the loss sequence using a Gaussian filter and identifies the stage transition points t_{early} and t_{late} according to the first- and second-order differences of the smoothed loss curve. The current stage is then determined by the position of the current round t relative to these two demarcation points. During the early stage, a lower pruning rate is used to retain more parameters and capture key data features; in the mid-term, the pruning rate is gradually increased to enhance the model's generalization ability and reduce the risk of overfitting; in the late stage, the highest pruning rate is applied to optimize the model structure and maximize efficiency. Each

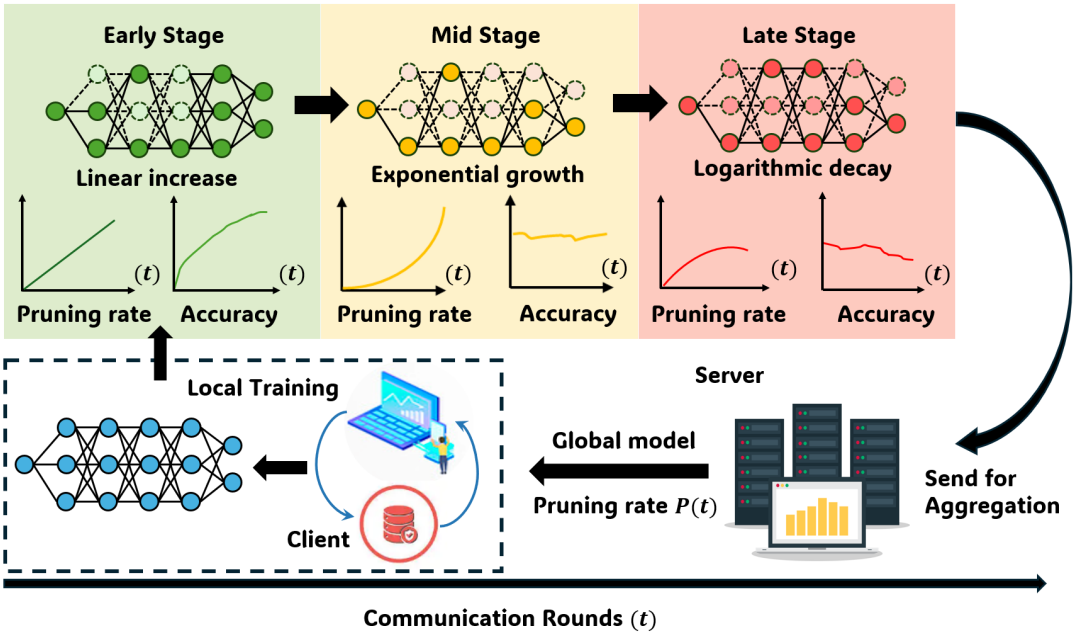


FIGURE 7.1. System architecture of the three-stage pruning strategy of FedSP.

client executes pruning locally according to the pruning strategy distributed by the server and then sends the updated model back to the server for aggregation. Through this strategy, segment-wise pruning not only significantly reduces the communication burden of the model but also enhances computational efficiency, making it a highly promising optimization method in FL.

7.2.2 Modeling

This approach is considered particularly effective because the model's tolerance for error and learning needs vary during different phases of training Weng *et al.* (2021); Zhang and Freris (2023). In addition, the minimum pruning rate $P_{\min}$ ensures that key features and weights are not discarded prematurely in the early stages of model training, thereby ensuring that the model has sufficient learning depth and quality. The maximum pruning rate $P_{\max}$ limits the upper limit of pruning to prevent damage to the model's generalization ability and overall performance due to excessive pruning in the middle and late stages of training. Figure 7.2 illustrates the changes in the pruning rate at different stages. Three distinct phases of training

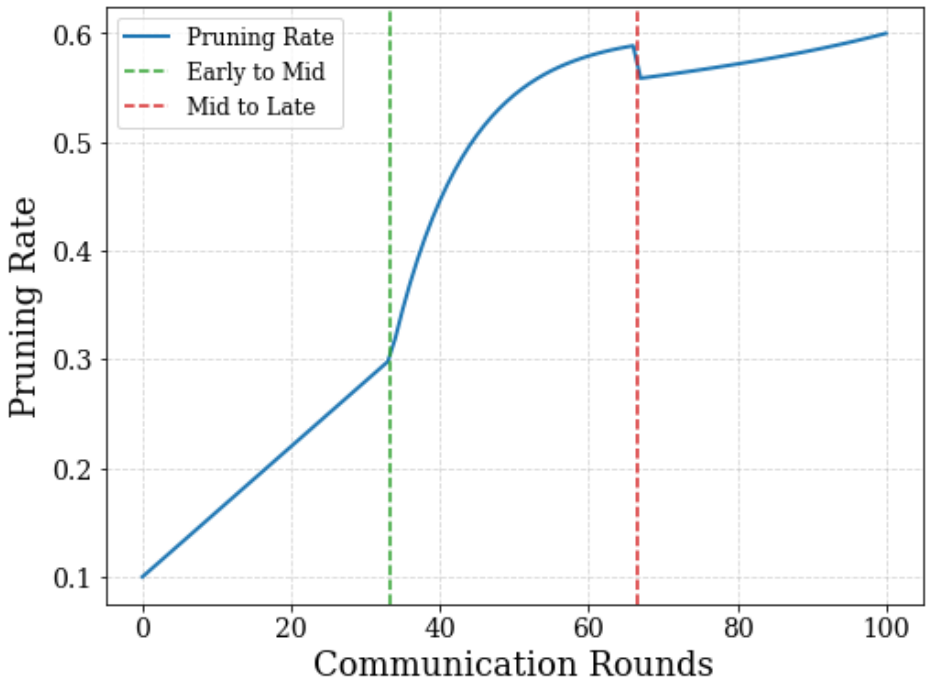


FIGURE 7.2. Dynamic pruning rate adjustment for the FedSP framework. The x-axis represents the communication rounds (t), while the y-axis shows the pruning rate ($P(t)$).

are illustrated:

Early Stage (green dashed line): Pruning rate linearly increases from $P_{\text{min}} = 0.1$ to $P_{\text{mid}} = 0.3$, enabling the model to retain essential parameters while exploring key features Joo *et al.* (2022).

Mid Stage (between green and red dashed lines): Pruning rate grows exponentially from $P_{\text{mid}} = 0.3$ to $P_{\text{max}} = 0.6$ with a growth rate of $\alpha = 0.1$, optimizing communication efficiency by aggressively pruning redundant parameters Li *et al.* (2025).

Late Stage (after red dashed line): Pruning rate decreases logarithmically from $P_{\text{max}} = 0.6$ with a decay factor of $\beta = 0.2$, stabilizing the model and preventing over-pruning Orseau *et al.* (2020). The total number of communication rounds is $T = 100$. These segmented strategies align pruning dynamics with the model's learning process, ensuring robust performance across diverse datasets.

In the early stages of training $0 \leq t < t_{\text{early}}$, the model's primary objective is to maximize information retention and feature extraction, as the parameters are yet to converge and may

contain essential representations for future learning. During this phase, the pruning rate must remain minimal to avoid premature removal of critical parameters. A linear growth strategy is adopted to ensure a smooth, incremental increase in the pruning rate, minimizing disruptions to the training dynamics while establishing a robust baseline structure for the model. The pruning rate $P(t)$ is defined as,

$$P(t) = P_{\min} + (P_{\text{mid}} - P_{\min}) \cdot \frac{3t}{T}, \tag{7.1}$$

where $P_{\min}$ denotes the initial pruning rate, $P_{\text{mid}} = \frac{P_{\min}+P_{\max}}{2}$ represents the intermediate pruning rate threshold, and T is the total number of training rounds. The linear growth ensures that the pruning process remains gradual and does not disrupt the convergence trajectory, thereby enabling the model to capture fundamental data characteristics without significant performance degradation.

In the mid-stage of training $t_{\text{early}} \leq t < t_{\text{late}}$, the model enters a critical phase where it begins optimizing its structure while mitigating the risk of overfitting. Overfitting, characterized by excellent performance on training data but poor generalization to unseen data, is addressed by exponentially increasing the pruning rate. This strategy rapidly eliminates redundant parameters, thereby reducing model complexity while preserving critical features contributing to generalization. The pruning rate $P(t)$ for this stage is defined as,

$$P(t) = P_{\text{mid}} + (P_{\max} - P_{\text{mid}}) \cdot \left(1 - e^{-\alpha \cdot (t - \frac{1}{3}T)}\right), \tag{7.2}$$

where α is a scaling parameter that controls the rate of exponential growth, and $P_{\max}$ is the maximum allowable pruning rate. Exponential growth aligns with the model's diminishing marginal gains during this phase, allowing for the efficient removal of low-impact parameters while maintaining sufficient capacity for critical features. This mechanism optimizes the trade-off between model size and generalization performance, accelerating convergence.

In the late stage of training $t_{\text{late}} \leq t < T$, the model stabilizes, and the training process focuses on fine-tuning its performance. During this phase, the primary concern is to avoid excessive pruning, which may disrupt the convergence of critical parameters and degrade overall performance. A logarithmic decay strategy is employed to achieve this, where the

pruning rate gradually decreases, enabling more refined adjustments to the model structure. The pruning rate $P(t)$ is expressed as,

$$P(t) = P_{\max} - (P_{\max} - P_{\text{mid}}) \cdot \beta \cdot \ln\left(1 + \frac{T-t}{\frac{T}{3}}\right), \tag{7.3}$$

where β is a modulation parameter that determines the logarithmic decay rate. This strategy ensures that the pruning process becomes less aggressive as training progresses, preserving critical features and maintaining the stability of the model's final performance. By carefully controlling pruning intensity, logarithmic decay enhances the robustness of the final model, particularly in FL scenarios where data distributions are heterogeneous and convergence stability is critical.

In **Algorithm** 6, we present the implementation of the proposed segmented pruning framework within the FL system. The framework systematically integrates the pruning strategies discussed for the early, mid, and late training stages. By dynamically adjusting the pruning rates based on the specific training phase, the framework ensures an effective balance between maintaining model accuracy and optimizing computational and communication efficiency.

Specifically, pruning is executed locally by each client according to the strategy distributed by the server. After local pruning, clients send the updated pruned models back to the server for aggregation. The server subsequently updates the global model and refines the pruning strategy based on performance feedback from the aggregated model. This iterative process achieves synchronized optimization of local and global models, ensuring robust training outcomes with reduced communication overhead.

To determine the critical demarcation points t_{early} and t_{late}, we analyze the rate of decline and its change over time in the training loss. A Gaussian smoothing filter is applied to reduce noise in the training loss sequence $L = \{l_1, l_2, \ldots, l_T\}$. The smoothed loss sequence $\hat{L} = \{\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_T\}$ is computed as described in **Algorithm** 8.

Importantly, the stage boundaries are not predefined or fixed for all system configurations. Instead, they are determined adaptively from the evolution of the global training loss. Since the loss dynamics reflect the overall convergence behavior of the federated model, this mechanism

Algorithm 6 Federated Segmented Pruning Framework

Input: Maximum pruning rate $P_{\max}$, Minimum pruning rate $P_{\min}$, Demarcation points $t_{\text{early}}, t_{\text{late}}$, Number of clients N, Communication rounds T, Learning rate η, Epoch count E

Output: Global model w^g

Initialize global model w_0^g
for each round $t = 1, 2, \ldots, T$ **do**
  for each client $n = 1, 2, \ldots, N$ **do**
    ClientUpdate($w_n^c(t)$) // local training and pruning
  end for
  Aggregation: update global model
  $w^g(t+1) \leftarrow \sum_{n=1}^{N} \frac{D^n}{D} w_k^c(t+1)$
end for
ClientUpdate($w_n^c(t)$)**:**
for each parameter $p \in w_n^c(t)$ **do**
  Compute pruning threshold $P(t)$ according to Eq.(7.1)–(7.3)
  Apply pruning to p based on $P(t)$
end for
for each local epoch $e = 1, 2, \ldots, E$ **do**
  $w_n^c(t+1) \leftarrow w_n^c(t) - \eta \cdot \nabla F_n(w_n^c(t))$
end for
Return $w_n^c(t+1)$

allows the stage segmentation to automatically adjust to different system setups, such as varying numbers of participating clients, datasets, or training durations.

In practice, the server monitors the global training loss at each communication round and applies Gaussian smoothing to reduce noise. The smoothed loss sequence is then used to compute the first- and second-order differences, which reveal the changes in the convergence speed. Based on these statistics, the server identifies the early-stage boundary t_{early} as the first inflection point where the loss decrease begins to slow down, and determines the late-stage boundary t_{late} when the loss change becomes stable. Once these boundaries are determined, the system can identify the current stage according to the current communication round t.

The early phase endpoint t_{early} is identified as the first point where the second-order difference of the loss ($\Delta_{l_t'}^2$) transitions from negative to positive,

$$t_{\text{early}} = \min \left\{ t \mid \Delta_{l_t'}^2 > 0, \forall t \in [3, T] \right\}. \tag{7.4}$$

Similarly, t_{late} is determined as the point where the second-order difference remains below a threshold ϵ, indicating convergence,

$$t_{\text{late}} = \min \left\{ t \mid \left| \Delta^2_{l'_t} \right| < \epsilon \cap t > t_{\text{early}} \right\}. \tag{7.5}$$

Algorithm 7 Determination of t_{early} and t_{late}

Input: Training loss sequence $L = \{l_1, l_2, \ldots, l_T\}$
Output: End point of the early phase t_{early} and the late phase t_{late}

1: Smooth the loss sequence using Algorithm 8
2: Compute the loss change rate Δ_L and its difference Δ^2_L
3: Determine t_{early} via Eq.(7.9) and t_{late} via Eq.(7.10)
4: **Return** t_{early}, t_{late}

Algorithm 8 Gaussian Smoothing

Input: Training loss sequence $L = \{l_1, l_2, \ldots, l_T\}$, window size N, standard deviation σ
Output: Smoothed sequence $\hat{L} = \{\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_T\}$

Compute $k \leftarrow \frac{N-1}{2}$ *(half-window size)*
Precompute Gaussian weights: $G(i) \leftarrow \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{i^2}{2\sigma^2}\right)$, for $i = -k, \ldots, k$
for each $t = 1, 2, \ldots, T$ **do**
    Initialize weighted sum $S \leftarrow 0$, weight sum $W \leftarrow 0$
    for each $i = -k$ to k **do**
        $j \leftarrow t + i$
        if $j \in [1, T]$ **then**
            $S \leftarrow S + G(i) \cdot l_j$
            $W \leftarrow W + G(i)$
        end if
    end for
    Compute smoothed value: $\hat{l}_t \leftarrow \frac{S}{W}$
end for
Return $\hat{L} = \{\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_T\}$

Note 1: When processing sequence data using Gaussian smoothing or any window-based smoothing method, mirror Padding is a choice to handle the boundary conditions for the data at the beginning and end of the sequence. Generally, $K = \frac{N-1}{2}$ filling points are required at the beginning and the end of the window, respectively. The padding sequence $\hat{L} = \{\hat{l'}_1, \hat{l'}_2, ..., \hat{l'}_{T+2K}\}$ can be described as **Algorithm** 7:

$$\hat{L} = \begin{cases} l'_t = l'_{-t+1}, & t = 1, 2, ..., k; \\ l'_t = l_t, & t = k+1, ..., T+k; \\ l'_{T+k+t} = l'_{T-t+1}, & t = 1, 2, ..., k. \end{cases} \tag{7.6}$$

Furthermore, we calculate the loss change rate at each point in the smoothed loss sequence $\hat{L}$ as follows:

$$\Delta_{l_t} = \hat{l}'_t - \hat{l}'_{t-1}, \tag{7.7}$$

where we can get a rate of change sequence $\Delta_L = \left\{\Delta_{l'_2}, \Delta_{l'_3}, ..., \Delta_{l'_T}\right\}$.

Ideally, we would like to find the first significant point where the rate of loss decrease slows down, which usually means the first inflection point of the loss curve, rather than simply finding a local minimum. In practice, since we are dealing with discrete data, this process can be approximated by looking at the continuous loss change rate differentials. If the difference in the rate of change of the loss changes from negative to positive, it means we have passed a local minimum point, but the real goal is to find where the loss starts to slow down. Considering this, we use the difference in the loss change rate instead of the loss change rate, and the difference of $\Delta_{l'_t}$ can be described as follows,

$$\Delta^2_{l'_t} = \Delta_{l'_t} - \Delta_{l'_{t-1}}, \tag{7.8}$$

where $t = 3, 4, ..., T$.

Therefore, the endpoint of the early phase t_{early} can be defined as the point in time when the difference in loss change rate first turns from negative to positive, indicating where the loss decline begins to slow down and can be defined as follows,

$$t_{\text{early}} = min\left\{t \middle| \left|\Delta^2_{l'_t}\right| > 0, \forall t \in [3, T]\right\}, \tag{7.9}$$

The endpoint of late phase t_{late} indicates the model's transition from faster learning to convergence or near-optimal performance. This usually means that the rate of loss change slows further or that the loss change starts to stabilize within a smaller range. The point t_{late} occurs when the difference in the loss change rate first sustains less than a given positive threshold

ϵ. This indicates that the learning rate of the model has started to stabilize. In addition, The point t_{late} should be after t_{early} and we can define it as,

$$t_{\text{late}} = min\left\{t| \left|\Delta^2_{l'_t}\right| < \epsilon \bigcap t > t_{\text{early}}]\right\}. \tag{7.10}$$

The proposed Federated Segmented Pruning (FedSP) framework focuses on the scheduling strategy of pruning rather than introducing a new pruning operator. In this chapter, we adopt magnitude-based weight pruning as the underlying pruning method, which is widely used due to its simplicity and effectiveness. Specifically, pruning is performed by removing parameters with small absolute weight values. For a given model parameter set p, a pruning rate $P(t)$ is first determined according to the current training stage. Based on this pruning rate, a threshold $\tau(t)$ is computed such that a proportion $P(t)$ of parameters with the smallest magnitudes are removed. Formally, the pruning operation can be expressed as

$$p' = p \odot M, \tag{7.11}$$

where M denotes the binary pruning mask and $\odot$ represents element-wise multiplication. The mask M is defined as

$$M_i = \begin{cases} 0, & |p_i| < \tau(t) \\ 1, & |p_i| \geq \tau(t) \end{cases}$$

where W_i is the i-th parameter of the model and $\tau(t)$ is the pruning threshold determined by the pruning rate $P(t)$.

In the federated learning process, pruning is executed locally on each client according to the pruning rate distributed by the server. After pruning and local training, the updated pruned model is transmitted to the server for aggregation. The proposed FedSP framework dynamically adjusts the pruning rate across different training stages, while the underlying pruning operation remains consistent across clients.

7.3 Theoretical Analysis

In this section, we provide theoretical guarantees for the proposed segmented pruning strategy in FL. We begin with auxiliary lemmas that describe the effect of pruning on optimization dynamics, and then build upon them to establish our main convergence results. Finally, we discuss the implications of pruning on communication efficiency and generalization under Non-IID data. All detailed proofs are deferred to the Appendix C.

LEMMA 7.1. ***Lyapunov Function Monotonicity.*** *Consider the Lyapunov function $V(w) = \|w - w^*\|^2$, where w^* is the global optimum. Under standard gradient descent updates, $V(w)$ decreases monotonically across iterations.*

Lemma 7.1 provides the basic stability tool we use in the following analysis. We next turn to the descent property of the global loss under pruning.

PROPOSITION 7.1. ***Global Loss Function Monotonicity with Pruning.*** *Let $F(w)$ be convex with Lipschitz continuous gradients. If the learning rate is chosen as*

$$\eta_t = \min\left(\frac{2(F(w_t)-F(w^*))}{\|\nabla F(w_t)\|^2},\ \frac{1}{L}\right),$$

then the global loss $F(w)$ decreases monotonically across iterations, even in the presence of pruning perturbations.

Having established monotonicity of the global loss, we next quantify the direct perturbation caused by pruning.

LEMMA 7.2. ***Impact of Pruning on Gradients and Loss.*** *Let $\mathcal{P}$ denote the pruning operator with pruning rate $P(t)$. Then the perturbation on gradients and the loss function is bounded by*

$$\|\nabla F(\mathcal{P}(w)) - \nabla F(w)\| \leq L \cdot P(t) \cdot \|w\|,$$

ensuring that pruning introduces limited and controllable errors.

We can also compare the original loss trajectory with the pruned one.

PROPOSITION 7.2. ***Impact of Pruning on Loss Function.*** *For a fixed number of communication rounds T, the loss of the pruned model, $F_p(w'_t)$, remains no larger than that of the unpruned model, $F(w_t)$, across training under appropriate pruning schedules.*

Having clarified the error bounds and loss behaviors, we now turn to the main convergence guarantees.

THEOREM 7.1. ***Local Convergence under Dynamic Pruning Rates.*** *During local updates at each client, the model parameters converge towards the local optimum. The convergence rate is preserved up to an additive error term explicitly controlled by the pruning rate $P(t)$.*

The above Theorem 7.1 ensures that pruning does not prevent clients from approaching their local optima. We next extend this analysis to the global aggregation process.

THEOREM 7.2. ***Global Convergence under Dynamic Pruning Rates.*** *During global aggregation, the federated model converges to a neighborhood of the global optimum. The error floor depends on the cumulative effect of the dynamic pruning rate $P(t)$, which remains bounded throughout training.*

Beyond convergence, pruning also influences the system-level behavior of FL. We first examine the pruning schedule itself.

REMARK 7.1. ***On Pruning-rate Schedules.*** *The proposed pruning schedule $P(t)$ is monotonic within each training stage (early, middle, late), ensuring gradual pruning and stable convergence.*

We then quantify the communication savings that pruning brings.

PROPOSITION 7.3. ***Communication Overhead Reduction.*** *Let p_t denote the proportion of parameters retained after pruning at round t. Then the communication cost per round is reduced by a factor proportional to $1 - p_t$, while the convergence error increases at most by $O(p_t^2)$.*

Finally, we analyze the generalization ability of the pruned model in heterogeneous environments.

THEOREM 7.3. ***Generalization under Non-IID Pruning.*** *By reducing model complexity, pruning improves the generalization ability of the global model. Under Non-IID data distributions, dynamic pruning rates $P(t)$ mitigate gradient variance across clients, thereby preserving generalization performance.*

These nine results show that segmented pruning in FL provides convergence guarantees at both local and global levels, reduces communication overhead, and enhances generalization under Non-IID data.

7.4 Experiments

7.4.1 Experimental Setup

To validate the effectiveness of the proposed FedSP framework in dynamic ICPS scenarios, we conducted extensive experiments under realistic and diverse conditions. The setup includes datasets, model architectures, hyperparameters, system configurations, and baseline methods, as detailed below.

7.4.2 Datasets

This chapter uses six public datasets: DermMNIST, Gas Sensor Array Drift Dataset (GSAD), State Farm Distracted Driver Detection (SFDD), CIFAR-10, CIFAR-100, and CelebA. These datasets cover multiple domains including medical image analysis, industrial sensor modeling, and driver behavior recognition. They also include both image and multidimensional time-series data, which allows a broad evaluation of the proposed model's generalization and robustness. The DermMNIST dataset comes from the MedMNIST v2 collection and focuses on skin lesion classification. It contains 10,015 RGB dermoscopic images sized 28×28 pixels, divided into training, validation, and testing sets. The dataset defines seven categories: 0 —

actinic keratoses and intraepithelial carcinoma, basal cell carcinoma, benign keratosis-like lesions, dermatofibroma, melanoma, melanocytic nevi, and vascular lesions. GSAD records data from a 16-sensor metal oxide semiconductor (MOS) array that monitored six types of gas mixtures over 36 months. Each sample includes 128 continuous features, and the dataset contains about 13,910 samples. It includes six gas mixture classes: Ammonia, Acetaldehyde, Acetone, Ethylene, Ethanol, and Toluene. SFDD provides in-car camera images (640×480 pixels) showing the driver's upper body and hands, with around 22,000 training images and 79,000 test images. The dataset defines ten driving behavior categories: safe driving,texting (right hand), talking on the phone (right hand), texting (left hand), talking on the phone (left hand), operating the radio, drinking, reaching behind, hair and makeup, and talking to passenger. To further test generalization on standard computer vision tasks, the study also includes CIFAR-10, CIFAR-100, and CelebA, which are well-known benchmarks for image classification and facial attribute recognition.

7.4.3 Model Architectures and Hyperparameters

Local model architectures are designed according to the characteristics and complexity of each dataset. As shown in Table 7.1, different neural network structures are adopted for different data modalities, including convolutional, fully connected, and deep pretrained networks. Table 7.2 summarizes the dataset-specific training hyperparameters.

7.4.4 System Configuration

We adopted six devices with different configurations to simulate realistic heterogeneous environments and device specifications were given in Table 7.3. We simulate Non-IID data using a Dirichlet distribution with parameter a. To emulate mild network fluctuations, we used the Linux tool tc netem to inject a 0.1% random packet-loss rate and a fixed 50 ms one-way delay into each client link. All devices were connected through a 100 Mbps wired Ethernet network, providing an effective throughput of approximately 93 Mbps. After delay injection, the average round-trip time (RTT) was about 105 ms.

TABLE 7.1. Architectures of local models.

Dataset / Model	Main Structure Description
DermMNIST (SimpleCNN)	Conv(3→32, k3, BN, ReLU), MaxPool Conv(32→64, k3, BN, ReLU), MaxPool Conv(64→128, k3, BN, ReLU), AdaptiveAvgPool2d(1) FC(128→7)
GSAD (MLP)	Input(128) → FC(128), ReLU, Dropout(0.2) FC(64), ReLU, Dropout(0.2) → FC(6)
SFDD (EfficientNet-B0)	Pretrained EfficientNet-B0 Backbone Classifier replaced with FC(num_features→10)
CIFAR-10 (VGG9)	Conv blocks: (32, 64, pool), (2×128, pool), (2×256, pool) FC(256, 256, 10)
CIFAR-100 (ResNet18)	64, pool; $2 \times [64, 64]$, $2 \times [128, 128]$, $2 \times [256, 256]$, $2 \times [512, 512]$ avgpool, FC(512→100)
CelebA (ResNet18)	Same as ResNet18

TABLE 7.2. Hyperparameter settings for datasets.

Dataset	Learning Rate η	Batch Size B	Epochs	Rounds
DermMNIST	0.005	64	3	150
GSAD	0.01	32	1	50
SFDD	0.001	32	30	5
CIFAR-10	0.001	128	5	200
CIFAR-100	0.001	128	5	200
CelebA	0.001	64	5	150

Communication was implemented via MPI over TCP (OpenMPI 4.1). Each training round used MPI_Bcast for model dissemination and MPI_Reduce for gradient aggregation.

TABLE 7.3. Configuration of server and client devices.

	CPU	GPU	VRAM
Server	i7-12700H @ 2.30 GHz	RTX 3070	64 GB
Client 1	i7-12700H @ 2.30 GHz	RTX 3070	32 GB
Client 2	i7-8565U @ 1.80 GHz	MX230	8 GB
Client 3	i9-13900HX @ 2.20 GHz	RTX 4060	16 GB
Client 4	i5-13500HX @ 4.70 GHz	RTX 4060	32 GB
Client 5	i5-10400F @ 2.90 GHz	RTX 3060	12 GB

7.4.5 Baselines and Metrics

We compare FedSP against state-of-the-art FL methods: FedAvg McMahan *et al.* (2017), Feddm Xiong *et al.* (2023), PHFL Pervej *et al.* (2024), AEDFL Liu *et al.* (2024a), PrunFL Jiang *et al.* (2022a), EFLMP Wu *et al.* (2023b), Fedduap Zhang *et al.* (2022a), FLASH Salehi *et al.* (2024), and FedMP Jiang *et al.* (2023).

7.4.6 Overall Comparison

To evaluate the overall performance, we compared FedSP with nine state-of-the-art (SOTA) FL algorithms across six benchmark datasets, including GSAD, SFDD, DermMNIST, CIFAR-10, CIFAR-100, and CelebA. These datasets cover diverse application domains ranging from medical imaging to natural image classification, allowing a comprehensive assessment under realistic non-IID settings (Dirichlet parameter $a = 0.2$). Here the pruning rate of FedSP is $P_{\text{min}} = 0.1$, $P_{\text{mid}} = 0.2$, $P_{\text{max}} = 0.4$) The quantitative results are presented in Table 7.4.

FedSP consistently achieved the highest accuracy among all compared methods. For instance, on the GSAD dataset, FedSP reached 88.9%, surpassing FedAvg (82.5%) and PHFL (81.6%). Similarly, it achieved 80.5% on SFDD and 85.4% on DermMNIST, outperforming other pruning-based methods such as PrunFL and AEDFL. In large-scale image classification tasks, including CIFAR-10 and CIFAR-100, FedSP achieved 74.8% and 71.6%, exceeding the performance of FedAvg by 12.7% and 15.8%, respectively. These results verify that FedSP effectively preserves critical parameters during adaptive pruning while mitigating performance degradation caused by data heterogeneity.

In terms of communication overhead, FedSP exhibits a substantial reduction compared to FedAvg across all datasets. For example, on GSAD, the total communication volume of FedSP was 3.31×10^2 MB, representing a 25% reduction relative to FedAvg (4.40×10^2 MB). Similarly, on SFDD and DermMNIST, FedSP achieved reductions of approximately 41.4% and 25.0%, respectively. Even on complex datasets such as CIFAR-10 and CIFAR-100, FedSP required only 2.16×10^4 MB and 2.64×10^4 MB, considerably less than FedAvg's 2.88×10^4 MB and 3.52×10^4 MB. This consistent advantage demonstrates that the stage-aware pruning

mechanism in FedSP efficiently compresses model updates, thereby lowering the transmission burden without affecting convergence.

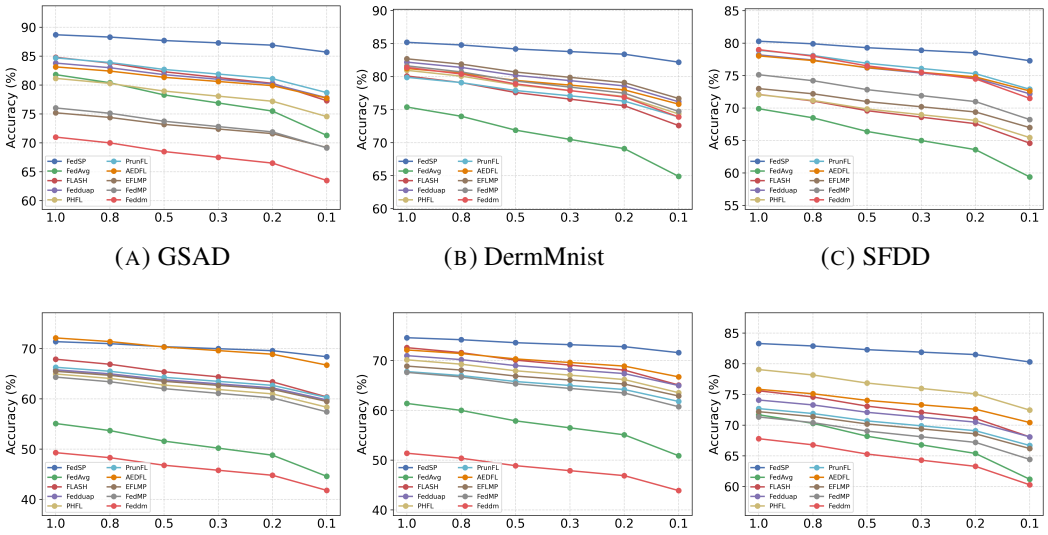
Since communication time directly reflects the overall transmission delay, it serves as a practical indicator of real-world efficiency. As shown in Table 7.4, FedSP achieved the shortest communication time in all experiments. For instance, it required only 2.84×10^1 s on GSAD, significantly faster than FedAvg's 3.78×10^1 s. On the larger SFDD dataset, FedSP reduced communication time from 4.48×10^3 s (FedAvg) to 3.36×10^3 s—a 25% improvement. Even in high-dimensional tasks such as CIFAR-100, FedSP achieved a communication time of 2.27×10^3 s, markedly shorter than competing methods (e.g., FLASH: 2.40×10^3 s, Fedduap: 2.29×10^3 s). These reductions confirm that FedSP not only lowers the data volume but also effectively accelerates the synchronization process between clients and server.

FedSP achieves higher predictive performance with the lowest communication cost and delay across all datasets. The combination of adaptive sparsification and progressive pruning enables FedSP to operate effectively under bandwidth-constrained and heterogeneous federated environments, providing a scalable and communication-efficient FL framework.

7.4.7 Robustness under data heterogeneity

To assess the robustness of FedSP underin the presence of data heterogeneity, we conducted experiments across a wide range of Non-IID levels controlled byevaluated its performance across a spectrum of Non-IID settings using the Dirichlet distribution parameter a. Smaller a values of a correspond to more skewed and, client-specific data distributions, simulareflecting real-world federated environments where client data can vary significantly in content and quantity. We evaluated six datasets—GSAD, SFDD, DermMNISTdeployment scenarios in industrial FL systems. The accuracy results under different a values are illustrated in Figure 7.3.

As expected, all methods experience a decreross a range of a values, from 0.1 (highly Non-IID) to values approaching IID conditions. Results are shown in Figure 7.3. FedSP consistently outperformed base lin accuracy as a decreases, reflecting the challenge of highly



(A) GSAD (B) DermMnist (C) SFDD

(D) CIFAR100 (E) CIFAR10 (F) CelebA

FIGURE 7.3. Accuracy of different methods under varying Non-IID degress on six datasets.

heterogeneous data. However, FedSP demonstrates consistently superior performance and slower degradation compared with all baselines. For instance, when a drops from 1.0 (near-IID) to 0.1 (highly Non-IID), the accuracy loss ofe methods across all levels of Non-IID. On the METR-LA dataset, for example, FedSP achieved 89.5% accuracy under strong Non-IID conditions ($a = 0.1$), compared to 73.1% for FedAvg and 80.4% for FedMP. Even under near-IID settings ($a \to \infty$), FedSP remtains within 4–6%, whereas FedAvg and Fedduap suffer reductions exceeding 10% and 8%, respectively. Similar trends are observed across all six datasets, confirming that FedSP generalizes effectively under diverse Non-IID conditions. The enhanced robustness of FedSP primarily stems from its stage-aware paraed its advantage, reaching 96.8% accuracy. Similar patterns were observed on CIFAR10 and other datasets, where the performance gap remained stable regardless of data skew.

7.4.8 Impact of pruning rate balance

To investigate how pruning intensity influences model learning and stability, we conducted ablation experiments under four distinct pruning configurations, each characterized by a specific combination of minimum, mid-stage, and maximum pruning rates ($P_{\text{min}}, P_{\text{mid}}, P_{\text{max}}$).

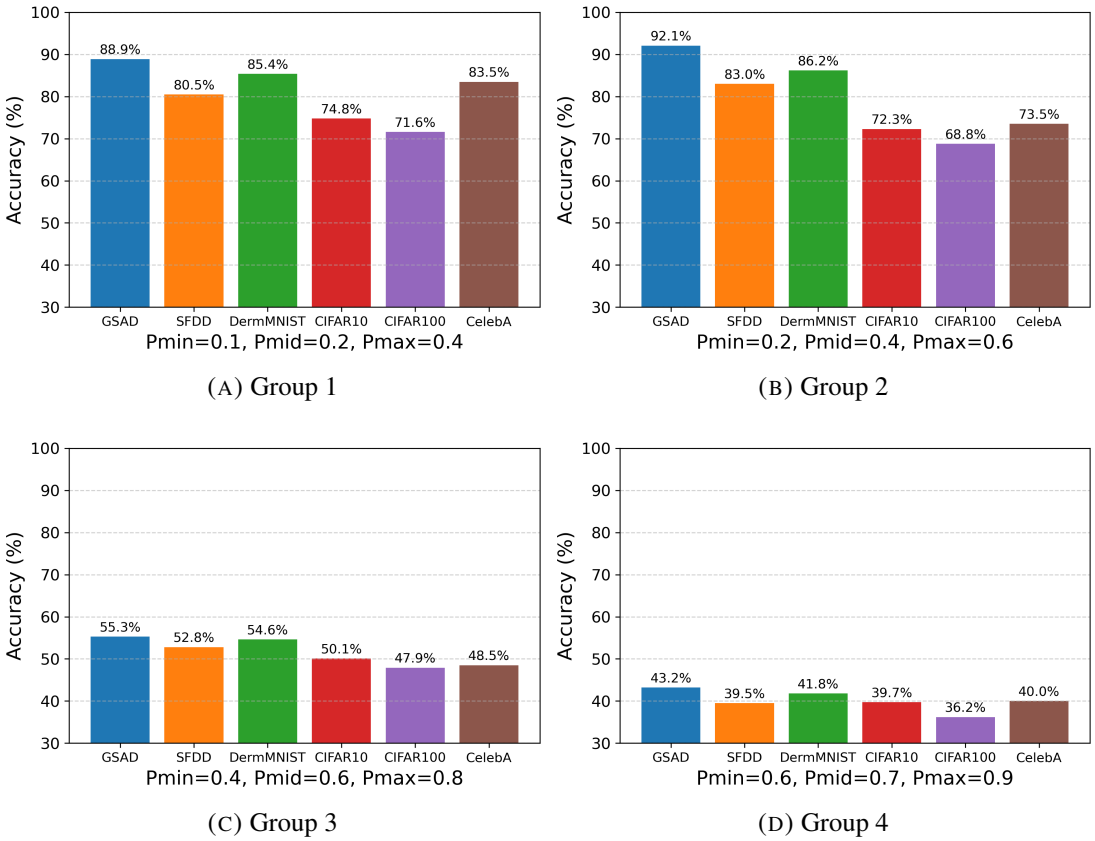
These configurations were evaluated on six datasets—GSAD, SFDD, DermMNIST, CIFAR-10, CIFAR-100, and CelebA—to analyze the trade-off between pruning aggressiveness and model accuracy. The results, summarized in Figure 7.4, demonstrate that excessive pruning significantly deteriorates performance, while a gradual and balanced pruning schedule yields the most reliable outcomes.

Groups 1 and 2, which employed conservative or moderately increasing pruning schedules, consistently achieved high accuracy across all datasets. For instance, both groups maintained accuracy above 90% on GSAD and SFDD, and above 75% on image classification tasks such as CIFAR-10 and CelebA. In contrast, Groups 3 and 4, which adopted aggressive pruning (e.g., P_{max}=0.8 or higher), experienced severe performance degradation, with accuracy dropping to around 50% and 40% respectively on the more complex datasets. This confirms that overly aggressive early-stage pruning removes critical parameters prematurely, hindering the model's ability to extract and retain discriminative features.

Among all configurations, Group 2 (P_{min}=0.2, P_{mid}=0.4, P_{max}=0.6) emerged as the most effective balance between sparsity and generalization. These findings underscore the importance of a **progressive, stage-aware pruning schedule** that adapts to model convergence dynamics. Rather than reacting to transient gradient variations, FedSP follows a predesigned, learning-phase-aligned pruning curve that preserves representational capacity early while promoting compression efficiency in later stages. This principled design enables FedSP to maintain high accuracy, strong robustness, and reduced communication overhead across diverse datasets and task domains.

7.4.9 Effectiveness of stage-aware pruning

To evaluate the effectiveness of the proposed three-stage pruning strategy in FedSP, we compared it with fixed-rate pruning and adaptive methods such as FedMP across six representative datasets: GSAD, SFDDD, DermMNIST, CIFAR10, CIFAR100, and CelebA. Two pruning configurations were tested: a moderate setting ($P_{\text{min}} = 0.1$, $P_{\text{mid}} = 0.2$, $P_{\text{max}} = 0.4$) and a more aggressive setting ($P_{\text{min}} = 0.2$, $P_{\text{mid}} = 0.4$, $P_{\text{max}} = 0.6$). As shown in Figure 7.5, the



(A) Group 1 (B) Group 2

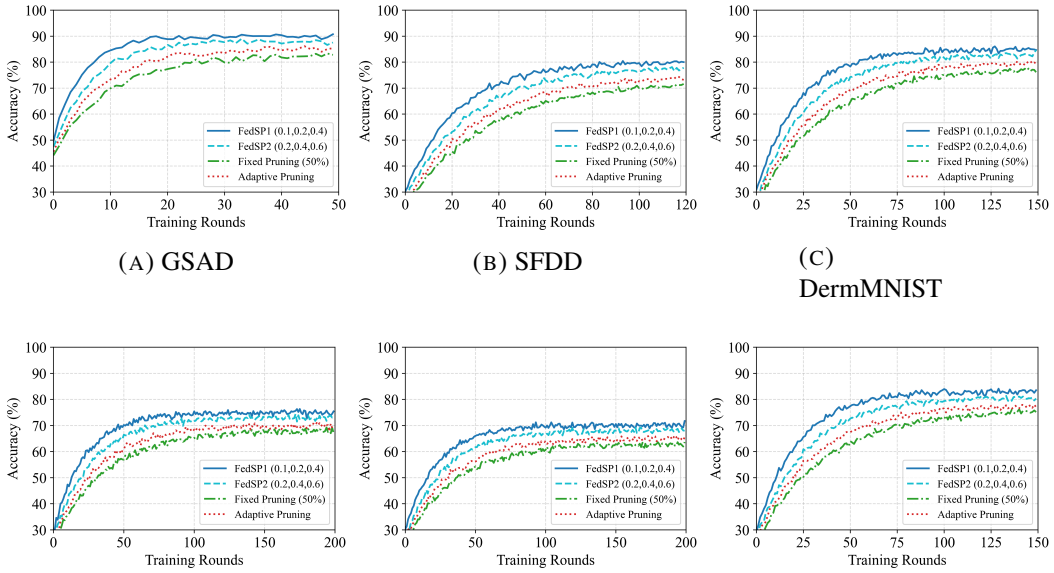
(C) Group 3 (D) Group 4

FIGURE 7.4. Accuracy trends across six datasets under four distinct dynamic pruning rate configurations, reflecting varying levels of pruning aggressiveness.

moderate configuration (FedSP1) consistently achieved higher accuracy and faster convergence across all datasets compared to both the aggressive configuration (FedSP2) and the baseline methods.

FedSP1 exhibited particularly stable behavior on complex visual datasets such as CIFAR100 and CelebA, where the aggressive pruning schedule of FedSP2 led to a noticeable drop in accuracy. The advantage of FedSP lies in its linear–exponential–logarithmic pruning profile, which preserves critical parameters during the early learning stage, accelerates convergence by pruning redundant weights in the middle stage, and maintains model generalization through conservative pruning in later rounds. In contrast, fixed-rate pruning (set at 50%) lacks sensitivity to model evolution, while adaptive methods such as FedMP may overreact to transient gradients or incur extra computational overhead for frequent evaluations.

These results demonstrate that the stage-aware pruning mechanism effectively balances learning stability and communication efficiency. By aligning pruning intensity with the model's training dynamics, FedSP avoids premature capacity loss and over-compression, enabling faster convergence and higher accuracy in FL under diverse data and model conditions.



(A) GSAD (B) SFDD (C) DermMNIST

(D) CIFAR10 (E) CIFAR100 (F) Celeba

FIGURE 7.5. Training accuracy trends for different pruning strategies (Fixed Pruning Rate = 50%, Adaptive Pruning (FedMP), and the proposed FedSP) across six datasets. The FedSP method adopts a three-stage pruning strategy with dynamic pruning rates, tested using two configurations: FedSP1 represents the pruning rate: $P_{\text{min}} = 0.1$, $P_{\text{mid}} = 0.2$, $P_{\text{max}} = 0.4$; and FedSP2 represents the pruning rate: $P_{\text{min}} = 0.2$, $P_{\text{mid}} = 0.4$, $P_{\text{max}} = 0.6$. The curves demonstrate that FedSP consistently achieves faster convergence and higher accuracy compared to Fixed Pruning and Adaptive Pruning across all datasets. This highlights the effectiveness of the proposed dynamic three-stage pruning strategy in balancing communication efficiency and model performance.

7.4.10 Training time under client heterogeneity

To evaluate the *system-level efficiency* of FedSP under heterogeneous client environments, We measured the average per-round local training time (in seconds) of each participating client and calculated the average speedup ratio ($\times$) compared with the standard FedAvg baseline. All experiments were conducted under heterogeneous client settings, where each client had distinct computational resources and non-identical data distributions. The evaluation was

performed across six datasets (GSAD, SFDD, DermMNIST, CIFAR-10, CIFAR-100, and CelebA) with corresponding model architectures, as summarized in Table 7.5.

Table 7.5 summarizes the local training time and corresponding speedup ratios of different methods under heterogeneous client conditions. Across all datasets, FedSP consistently exhibits a noticeable improvement in computational efficiency compared with both classical (FedAvg, FLASH) and pruning-based methods (PrunFL, FedMP). The average per-round local training time of FedSP is reduced by 30–40% relative to FedAvg, resulting in an overall speedup of **1.47–1.61**$\times$. This trend holds consistently across all six datasets, confirming the scalability of the segmented pruning strategy. For lightweight architectures such as MLP on GSAD, FedSP achieves moderate gains (1.48$\times$), while for deep CNN-based models like EfficientNet-B0 and ResNet18, the improvement becomes more pronounced (up to 1.61$\times$). This indicates that the computational benefit of FedSP grows with model depth and parameter scale, where pruning-induced sparsity yields larger reductions in floating-point operations. Compared with other pruning-aware frameworks (e.g., PrunFL and AEDFL), FedSP delivers higher efficiency despite similar pruning ratios. This advantage arises from its *segmentation-aware design*, which dynamically adjusts pruning intensity in different training stages, allowing early-stage pruning to accelerate convergence and late-stage recovery to prevent underfitting. Moreover, FedSP maintains stable efficiency across heterogeneous clients (Client1–Client5), demonstrating its robustness to variations in computational capacity and data heterogeneity. These results demonstrate that FedSP not only maintains model accuracy and communication efficiency but also significantly enhances computational efficiency, making it well-suited for deployment in heterogeneous and resource-constrained federated environments.

7.5 Conclusion

This chapter introduced Federated Segmented Pruning (FedSP), a stage-aware pruning framework designed to enhance the efficiency and adaptability of FL in highly heterogeneous environments. Unlike conventional pruning strategies that rely on fixed or externally adaptive

rates, FedSP exploits the model's intrinsic stage-dependent sensitivity to pruning, dividing the training process into multiple phases and dynamically adjusting pruning intensity according to the model's convergence state. By coupling pruning schedules with the model's training dynamics, FedSP enables implicit adaptation to communication, system, and data heterogeneity—without requiring real-time network monitoring or additional coordination overhead. This design extends neural network pruning from a static compression technique to a dynamic system-level optimization mechanism, achieving balanced trade-offs between model sparsity, convergence stability, and overall system efficiency. FedSP provides a unified perspective on FL under compounded heterogeneity, offering methodological insights for developing scalable and robust FL frameworks in complex distributed systems.

TABLE 7.4. Performance evaluation of FedSP and baseline methods on six datasets. We compared our FedSP with other nine SOTA methods in terms of Accuracy(%), Communication overhead(MB) and Communication time(s).

Accuracy (%)	**FedSP**	FedAvg	FLASH	Fedduap	PHFL
GSAD	88.9	82.5	85.3	84.2	81.6
SFDD	80.5	70.6	72.6	78.6	72.5
DermMNIST	85.4	76.1	80.64	82.6	81.4
CIFAR-10	74.8	62.1	73.1	71.4	70.6
CIFAR-100	71.6	55.8	68.4	66.2	65.4
CelebA	83.5	72.4	76.1	74.5	79.5
Accuracy (%)	PrunFL	AEDFL	EFLMP	FedMP	Feddm
GSAD	85.1	83.5	75.6	76.54	71.5
SFDD	79.3	78.4	73.4	75.6	79.5
DermMNIST	80.3	81.6	83.1	82.1	81.9
CIFAR-10	68.2	72.5	69.3	68.1	51.9
CIFAR-100	66.7	72.5	65.9	64.8	49.8
CelebA	73.1	76.2	72.6	71.8	68.3
Com. (MB)	**FedSP**	FedAvg	FLASH	Fedduap	PHFL
GSAD	$\mathbf{3.31\times10^2}$	4.40×10^2	3.71×10^2	3.74×10^2	3.35×10^2
SFDD	$\mathbf{3.05\times10^4}$	5.21×10^4	3.69×10^4	3.56×10^4	3.30×10^4
DermMNIST	$\mathbf{6.79\times10^3}$	9.05×10^3	8.42×10^3	8.38×10^3	7.43×10^3
CIFAR-10	$\mathbf{2.16\times10^4}$	2.88×10^4	2.73×10^4	2.22×10^4	2.27×10^4
CIFAR-100	$\mathbf{2.64\times10^4}$	3.52×10^4	2.79×10^4	2.66×10^4	2.78×10^4
CelebA	$\mathbf{1.83\times10^4}$	2.44×10^4	1.99×10^4	2.39×10^4	2.41×10^4
Com. (MB)	PrunFL	AEDFL	EFLMP	FedMP	Feddm
GSAD	3.56×10^2	4.03×10^2	4.24×10^2	4.26×10^2	4.33×10^2
SFDD	3.59×10^4	5.13×10^4	4.98×10^4	4.55×10^4	3.49×10^4
DermMNIST	8.96×10^3	6.91×10^3	7.01×10^3	7.74×10^3	8.97×10^3
CIFAR-10	2.35×10^4	2.35×10^4	2.19×10^4	2.70×10^4	2.63×10^4
CIFAR-100	2.65×10^4	2.78×10^4	2.83×10^4	2.71×10^4	2.64×10^4
CelebA	2.29×10^4	2.16×10^4	2.00×10^4	1.99×10^4	2.22×10^4
Time (s)	**FedSP**	FedAvg	FLASH	Fedduap	PHFL
GSAD	$\mathbf{2.84\times10^1}$	3.78×10^1	3.19×10^1	3.22×10^1	2.88×10^1
SFDD	$\mathbf{3.36\times10^3}$	4.48×10^3	3.17×10^3	3.06×10^3	2.84×10^3
DermMNIST	$\mathbf{5.83\times10^2}$	7.78×10^2	7.25×10^2	7.21×10^2	6.39×10^2
CIFAR-10	$\mathbf{1.86\times10^3}$	2.48×10^3	2.34×10^3	1.91×10^3	1.95×10^3
CIFAR-100	$\mathbf{2.27\times10^3}$	3.03×10^3	2.40×10^3	2.29×10^3	2.39×10^3
CelebA	$\mathbf{1.58\times10^3}$	2.10×10^3	1.72×10^3	2.06×10^3	2.07×10^3
Time (s)	PrunFL	AEDFL	EFLMP	FedMP	Feddm
GSAD	3.06×10^1	3.46×10^1	3.65×10^1	3.67×10^1	3.72×10^1
SFDD	3.08×10^3	4.42×10^3	4.28×10^3	3.91×10^3	3.00×10^3
DermMNIST	7.71×10^2	5.94×10^2	6.03×10^2	6.66×10^2	7.71×10^2
CIFAR-10	2.02×10^3	2.02×10^3	1.88×10^3	2.32×10^3	2.26×10^3
CIFAR-100	2.28×10^3	2.39×10^3	2.43×10^3	2.33×10^3	2.27×10^3
CelebA	1.97×10^3	1.86×10^3	1.72×10^3	1.71×10^3	1.91×10^3

TABLE 7.5. Average per-round local training time (in seconds) and average speedup ratios (×) of different FL methods across six datasets under heterogeneous client settings.

GSAD (MLP)										
Client	FedAvg	FedSP	FLASH	Fedduap	PHFL	PrunFL	AEDFL	EFLMP	FedMP	FedDM
Client1	0.54	0.33	0.53	0.52	0.46	0.48	0.44	0.47	0.52	0.49
Client2	17.00	11.60	15.26	16.48	14.24	16.94	15.22	16.00	16.66	15.51
Client3	0.73	0.49	0.73	0.71	0.67	0.66	0.65	0.63	0.72	0.75
Client4	0.73	0.50	0.72	0.64	0.67	0.66	0.65	0.63	0.72	0.68
Client5	0.87	0.54	0.86	0.77	0.72	0.87	0.70	0.74	0.84	0.89
Ratio	1.00	**1.48**	1.10	1.04	1.19	1.01	1.13	1.08	1.02	1.08
SFDD (EfficientNet-B0)										
Client	FedAvg	FedSP	FLASH	Fedduap	PHFL	PrunFL	AEDFL	EFLMP	FedMP	FedDM
Client1	302.80	188.51	299.54	291.50	281.62	275.97	242.78	258.84	294.85	280.96
Client2	9462.50	6454.49	8519.17	9154.18	7942.09	9406.21	7685.89	8929.50	8307.17	9611.34
Client3	406.99	280.09	364.39	359.25	378.80	363.97	331.48	348.37	361.65	415.27
Client4	406.99	279.95	399.21	394.24	373.43	402.83	326.69	347.16	363.30	415.68
Client5	485.26	302.58	480.89	423.94	445.44	435.82	434.21	412.26	430.08	491.55
Ratio	1.00	**1.47**	1.10	1.04	1.17	1.02	1.23	1.07	1.13	0.99
DermMNIST (SimpleCNN)										
Client	FedAvg	FedSP	FLASH	Fedduap	PHFL	PrunFL	AEDFL	EFLMP	FedMP	FedDM
Client1	83.16	56.92	82.60	72.94	69.32	75.02	66.71	79.15	80.68	76.44
Client2	2598.75	1598.54	2335.20	2533.28	2414.89	2579.57	2097.68	2218.20	2552.29	2658.30
Client3	111.77	68.65	98.03	97.97	103.54	110.59	99.45	105.06	99.78	113.76
Client4	111.77	69.52	99.23	107.61	93.38	110.62	90.45	104.69	99.82	113.03
Client5	133.27	90.89	130.85	116.06	112.08	119.59	108.22	125.94	131.93	122.76
Ratio	1.00	**1.61**	1.11	1.04	1.09	1.01	1.23	1.15	1.02	0.99
CIFAR-10 (VGG9)										
Client	FedAvg	FedSP	FLASH	Fedduap	PHFL	PrunFL	AEDFL	EFLMP	FedMP	FedDM
Client1	85.60	52.48	84.46	82.99	71.53	84.89	68.52	80.24	84.03	87.49
Client2	2675.00	1816.46	2405.18	2349.84	2233.12	2673.84	2151.46	2273.06	2621.06	2449.99
Client3	115.05	71.52	102.80	100.06	96.89	104.82	102.88	98.88	111.65	117.11
Client4	115.05	78.10	103.26	100.24	106.98	102.75	102.02	109.63	113.19	107.06
Client5	137.18	85.44	134.30	132.41	127.96	137.87	122.57	117.16	132.88	139.90
Ratio	1.00	**1.49**	1.10	1.13	1.19	1.01	1.23	1.17	1.02	1.08
CIFAR-100 (ResNet18)										
Client	FedAvg	FedSP	FLASH	Fedduap	PHFL	PrunFL	AEDFL	EFLMP	FedMP	FedDM
Client1	85.60	58.08	76.40	74.55	79.01	77.96	68.67	72.64	75.95	79.34
Client2	2675.00	1657.52	2390.81	2347.02	2451.15	2689.28	2374.67	2310.31	2359.51	2704.17
Client3	115.05	70.35	102.00	101.73	107.31	102.83	92.28	108.39	101.77	106.49
Client4	115.05	78.42	113.90	111.90	95.66	104.52	103.24	108.71	112.19	105.04
Client5	137.18	84.99	122.42	120.42	113.95	137.54	110.58	129.02	133.48	138.93
Ratio	1.00	**1.60**	1.11	1.13	1.10	1.00	1.14	1.15	1.12	1.00
CelebA (ResNet18)										
Client	FedAvg	FedSP	FLASH	Fedduap	PHFL	PrunFL	AEDFL	EFLMP	FedMP	FedDM
Client1	94.40	58.02	83.83	82.10	79.71	94.07	83.57	89.66	84.05	96.57
Client2	2950.00	1829.40	2617.67	2872.14	2468.10	2685.56	2401.65	2765.38	2629.00	2726.88
Client3	126.88	78.82	114.21	122.76	118.15	125.79	102.27	108.57	123.16	116.95
Client4	126.88	86.67	114.47	122.68	116.92	115.52	112.98	119.90	113.09	117.96
Client5	151.28	92.54	134.70	145.63	138.91	152.05	135.29	128.63	148.31	154.06
Ratio	1.00	**1.61**	1.13	1.03	1.18	1.09	1.22	1.07	1.11	1.07

CHAPTER 8

Conclusion

Federated learning (FL), as a privacy-preserving paradigm for distributed model training, faces substantial challenges in real-world deployment. Communication constraints, inconsistent data distributions, and disparities in system capabilities jointly hinder its scalability and robustness. Most existing research focuses on optimizing one dimension of heterogeneity in isolation, lacking a unified theoretical formulation or systematic solution that can address multiple types of heterogeneity concurrently. This dissertation bridges that gap by proposing a unified mathematical framework and progressively validating it through a series of increasingly complex experimental settings. The work ultimately establishes a pruning-based, system-level optimization framework that completes the transition from algorithmic feasibility to practical applicability.

Chapter 2 laid the theoretical foundation by formalizing the global optimization problem of FL under heterogeneous conditions. In this unified formulation, communication, statistical, and system heterogeneity were jointly modeled within a single constrained objective, providing both a mathematical basis and a system-level perspective for the subsequent studies.

Chapter 4 conducted a proof-of-concept investigation focusing on communication heterogeneity. A server-side pruning algorithm based on parameter similarity was proposed to demonstrate that pruning can effectively reduce communication overhead without compromising model accuracy. This stage, serving as the methodological prototype, verified the feasibility of neural network pruning within standard FL systems and provided a methodological foundation for further extensions.

Chapter 5 scaled the research to a larger federated environment, examining communication efficiency when the number of participating clients increases significantly. Using **intelligent transportation systems (ITS)** as a representative application scenario, we introduced a two-stage pruning algorithm, **FEDCG**, which performs pruning on both client and server sides. By jointly optimizing the upload and download communication processes, FEDCG achieves substantial reductions in communication volume while preserving convergence stability and model accuracy. This work validates the scalability and adaptability of pruning-based optimization in large-scale federated systems.

Chapter 6 extended the study to data heterogeneity, focusing on the practically relevant phenomenon of **slightly skewed labels**. To address this intermediate form of statistical heterogeneity, we proposed a Gaussian mixture-based clustering and oversampling method (GMCFL), which effectively mitigates the adverse effects of mild label imbalance on model aggregation and generalization. This study complements the communication-oriented optimization of the earlier chapters and further demonstrates the compatibility of pruning-inspired strategies with heterogeneous data distributions.

Chapter 7 advanced the investigation to compounded heterogeneous environments, where communication, data, and system heterogeneity coexist. We introduced **Federated Segmented Pruning (FedSP)**, a stage-aware pruning framework that dynamically adjusts pruning intensity in coordination with the model's convergence state. FedSP achieves fine-grained balance across training stages, attaining up to **40% reduction in communication cost** and **8% improvement in model accuracy** compared with existing baselines. These results confirm that pruning remains stable and effective even under the most challenging federated settings, demonstrating its system-level applicability.

Our work fulfills its initial research objective — to construct and validate a unified pruning-based optimization framework that enhances the efficiency and robustness of federated learning under multidimensional heterogeneity. From theoretical formulation to empirical verification, and from communication efficiency to system-level optimization, this work reveals that pruning is not merely a compression technique but a **bridging mechanism**

between algorithmic design and system performance. It provides a feasible and theoretically grounded pathway for the scalable deployment of intelligent federated systems in practice.

In future work, we plan to extend the proposed pruning framework to support **dynamic client participation**. In particular, adaptive client selection and online scheduling mechanisms will be explored to dynamically adjust the pruning schedule and aggregation weights according to client availability, communication latency, and bandwidth conditions. Reinforcement learning or adaptive optimization strategies may also be investigated to enable the system to automatically determine appropriate pruning levels during training.

We also aim to develop **multi-level adaptive pruning strategies** that jointly optimize model structure and system resources. This may involve combining structured pruning with resource-aware optimization techniques to dynamically adjust model sparsity according to device computation capability, memory constraints, and network conditions. Such strategies could enable the framework to balance computation cost, communication latency, and model accuracy during training.

In addition, practical deployment considerations will be investigated to improve the applicability of the proposed framework in real-world systems. Future work will examine the compatibility of server-side pruning with commonly used FL security mechanisms such as **secure aggregation** and **differential privacy**, as well as its robustness under potential adversarial or unreliable client behaviors. Addressing these practical challenges will further strengthen the deployment readiness of the proposed approach.

By pursuing these directions, we aim to advance federated learning from a communication-efficient and privacy-preserving paradigm toward a truly **system-optimized and resource-adaptive intelligent framework**, capable of supporting large-scale, real-world deployments across diverse heterogeneous environments.

Bibliography

Ahmed M Abdelmoniem, Chen-Yu Ho, Pantelis Papageorgiou, and Marco Canini. A comprehensive empirical study of heterogeneity in federated learning. *IEEE Internet of Things Journal*, 10(16):14071–14083, 2023.

Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. Privacy and human behavior in the age of information. *Science*, 347(6221):509–514, 2015.

Accountability Act et al. Health insurance portability and accountability act of 1996. *Public law*, 104:191, 1996.

Arshia Aflaki, Hadis Karimipour, and Thippa Reddy Gadekallu. Privacy-prioritized model aggregation in icps: A novel approach to federated learning aggregation with lime and blockchain. *IEEE Transactions on Industrial Cyber-Physical Systems*, 2024.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 440–445. Association for Computational Linguistics, 2017.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1709–1720, 2017.

Mohamad Arafeh, Ahmad Hammoud, Hadi Otrok, Azzam Mourad, Chamseddine Talhi, and Zbigniew Dziong. Independent and identically distributed (iid) data assessment in federated learning. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 293–298. IEEE, 2022.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

Lu Bai, Ziwei Huang, Yuming Ge, Rundong Yu, Longxiang Wang, and Xiang Cheng. Cellular vehicle-to-everything (c-v2x) testing: From theory to practice. *IEEE Network*, 2025.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 560–569. PMLR, 2018.

Rajendra Bhatia and Chandler Davis. A cauchy-schwarz inequality for operators with applications. *Linear algebra and its applications*, 223:119–129, 1995.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

Cheng Chen, Ziyi Chen, Yi Zhou, and Bhavya Kailkhura. Fedcluster: Boosting the convergence of federated learning via cluster-cycling. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5017–5026. IEEE, 2020.

Gaode Chen, Xinghua Zhang, Yijun Su, Yantong Lai, Ji Xiang, Junbo Zhang, and Yu Zheng. Win-win: A privacy-preserving federated framework for dual-target cross-domain recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4149–4156, 2023.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Derui Ding, Qing-Long Han, Xiaohua Ge, Xian-Ming Zhang, and Jun Wang. Privacy-preserving filtering, control and optimization for industrial cyber-physical systems. *Science*

China Information Sciences, 68(4):1–17, 2025.

Jun Du, Bingqing Jiang, Chunxiao Jiang, Yuanming Shi, and Zhu Han. Gradient and channel aware dynamic scheduling for over-the-air computation in federated edge learning systems. *IEEE Journal on Selected Areas in Communications*, 41(4):1035–1050, 2023.

Jian-hui Duan, Wenzhong Li, Derun Zou, Ruichen Li, and Sanglu Lu. Federated learning with data-agnostic distribution fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8074–8083, 2023.

Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201, 2009.

Hugh Everett III. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations research*, 11(3):399–417, 1963.

Farzan Farnia, Amirhossein Reisizadeh, Ramtin Pedarsani, and Ali Jadbabaie. An optimal transport approach to personalized federated learning. *IEEE Journal on Selected Areas in Information Theory*, 3(2):162–171, 2022.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

Jack Goetz, Kshitiz Malik, Duc Bui, Seungwhan Moon, Honglei Liu, and Anuj Kumar. Active federated learning. *arXiv preprint arXiv:1909.12641*, 2019.

Eric Goldman. An introduction to the california consumer privacy act (ccpa). *Santa Clara Univ. Legal Studies Research Paper*, 2020.

AA Goldstein. Optimization of lipschitz continuous functions. *Mathematical Programming*, 13:14–22, 1977.

Geoff Gordon and Ryan Tibshirani. Karush-kuhn-tucker conditions. *Optimization*, 10(725/36):725, 2012.

William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6):789–828, 1996.

Bowen Gu, Dong Li, Haiyang Ding, Gongpu Wang, and Chintha Tellambura. Breaking the interference and fading gridlock in backscatter communications: State-of-the-art, design challenges, and future directions. *IEEE Communications Surveys & Tutorials*, 2024.

Souhila Badra Guendouzi, Samir Ouchani, and Mimoun Malki. Aggregation using genetic algorithms for federated learning in industrial cyber-physical systems. In *2022 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6. IEEE, 2022.

Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *International Conference on Machine Learning*, pages 3973–3983. PMLR, 2020.

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

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

David Hästbacka, Jari Halme, Laurentiu Barna, Henrikki Hoikka, Henri Pettinen, Martin Larrañaga, Mikael Björkbom, Heikki Mesiä, Antti Jaatinen, and Marko Elo. Dynamic edge and cloud service integration for industrial iot and production monitoring applications of industrial cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 18(1):498–508, 2021.

Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(5):2900–2919, 2023.

Clare Elizabeth Heinbaugh, Emilio Luz-Ricca, and Huajie Shao. Data-free one-shot federated learning under very high statistical heterogeneity. In *The Eleventh International Conference on Learning Representations*, 2023.

Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. Distributed pruning towards tiny neural networks in federated learning. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pages 190–201. IEEE, 2023.

Divyansh Jhunjhunwala, Advait Gadhikar, Gauri Joshi, and Yonina C. Eldar. Adaptive quantization of model updates for communication-efficient federated learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3110–3114, 2021.

Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Zhida Jiang, Yang Xu, Hongli Xu, Zhiyuan Wang, Chunming Qiao, and Yangming Zhao. Fedmp: Federated learning through adaptive model pruning in heterogeneous edge computing. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 767–779. IEEE, 2022.

Zhida Jiang, Yang Xu, Hongli Xu, Zhiyuan Wang, Jianchun Liu, Qian Chen, and Chunming Qiao. Computation and communication efficient federated learning with adaptive model pruning. *IEEE Transactions on Mobile Computing*, 2023.

Yuchen Jiang, Jilun Tian, Shimeng Wu, Hao Luo, and Tianyi Gao. Security and privacy in industrial cyber-physical systems: Concerns, challenges, and countermeasures. In *Cyber Physical System 2.0*, pages 221–252. CRC Press, 2025.

Xiating Jin, Jiajun Bu, Zhi Yu, Hui Zhang, and Yaonan Wang. Fedcrack: Federated transfer learning with unsupervised representation for crack detection. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

Jiong Jin, Zhibo Pang, Jonathan Kua, Quanyan Zhu, Karl H Johansson, Nikolaj Marchenko, and Dave Cavalcanti. Cloud-fog automation: The new paradigm towards autonomous industrial cyber-physical systems. *IEEE Journal on Selected Areas in Communications*, 2025.

Donggyu Joo, Doyeon Kim, Eojindl Yi, and Junmo Kim. Linear combination approximation of feature for channel pruning. In *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition, pages 2772–2781, 2022.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

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

Sung Hoo Kim and Patricia L Mokhtarian. Finite mixture (or latent class) modeling in transportation: Trends, usage, potential, and future directions. *Transportation Research Part B: Methodological*, 172:134–173, 2023.

Minjae Kim, Sangyoon Yu, Suhyun Kim, and Soo-Mook Moon. Depthfl: Depthwise federated learning for heterogeneous clients. In *The Eleventh International Conference on Learning Representations*, 2022.

Minsu Kim, Walid Saad, Merouane Debbah, and Choong S Hong. Spafl: Communication-efficient federated learning with sparse models and low computational overhead. *Advances in Neural Information Processing Systems*, 37:86500–86527, 2024.

Hemant H Kumar, VR Karthik, and Mydhili K Nair. Federated k-means clustering: A novel edge ai based approach for privacy preservation. In *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 52–56. IEEE, 2020.

Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 19–35. USENIX Association, 2021.

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

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2018.

Vili Lehdonvirta, Bóxī Wú, and Zoe Hawkins. Compute north vs. compute south: the uneven possibilities of compute-based ai governance around the globe. In *Proceedings of the*

AAAI/ACM Conference on AI, Ethics, and Society, volume 7, pages 828–838, 2024.

Yun Li, Luyang Wang, Sifan Peng, Aakash Kumar, and Baoqun Yin. Using feature entropy to guide filter pruning for efficient convolutional networks. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Deep Learning: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part II 28*, pages 263–274. Springer, 2019.

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

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021. Commonly cited in 2022 works; provides standardized non-IID partitions and extensive evaluations.

Qinbin Li, Bingsheng He, and Dawn Song. FedBN: Federated learning on non-iid features via local batch normalization. In *International Conference on Learning Representations (ICLR)*, 2021.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 965–978. IEEE, 2022.

Wei Li, Jinlin Chen, Zhenyu Wang, Zhidong Shen, Chao Ma, and Xiaohui Cui. Ifl-gan: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10502–10515, 2022.

Qinbin Li, Bingsheng He, and Dawn Song. Adversarial collaborative learning on non-iid features. In *International Conference on Machine Learning*, pages 19504–19526. PMLR, 2023.

Jiachen Li, Yuchao Zhang, Yiping Li, Xiangyang Gong, and Wendong Wang. Fedsparse: A communication-efficient federated learning framework based on sparse updates. *Electronics*, 13(24):5042, 2024.

Lyu Li, Zekun Niu, Hang Yang, Junzhe Xiao, Guozhi Xu, Weisheng Hu, and Lilin Yi. Low complexity exponential pruning learned digital back-propagation method for fiber

nonlinearity mitigation. *Optics Letters*, 50(6):1893–1896, 2025.

Youzao Lian, Peng Peng, Kai Jiang, and Weisheng Xu. Cross-layer importance evaluation for neural network pruning. *Neural Networks*, 179:106496, 2024.

Dongping Liao, Xitong Gao, Yiren Zhao, and Cheng-Zhong Xu. Adaptive channel sparsity for federated learning under system heterogeneity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20432–20441, 2023.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations (ICLR)*, 2018.

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

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Su Liu, Jiong Yu, Xiaoheng Deng, and Shaohua Wan. Fedcpf: An efficient-communication federated learning approach for vehicular edge computing in 6g communication networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1616–1629, 2021.

Linfeng Liu, Zhiyuan Xi, Kun Zhu, Ran Wang, and Ekram Hossain. Mobile charging station placements in internet of electric vehicles: A federated learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):24561–24577, 2022.

Ji Liu, Tianshi Che, Yang Zhou, Ruoming Jin, Huaiyu Dai, Dejing Dou, and Patrick Valduriez. Aedfl: efficient asynchronous decentralized federated learning with heterogeneous devices. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pages 833–841. SIAM, 2024.

Yufeng Liu, Sheng Wang, Jixin Dong, Linghui Chen, Xinyu Wang, Lei Wang, Fudong Li, Chenchen Wang, Jiahai Zhang, Yuzhu Wang, et al. De novo protein design with a denoising diffusion network independent of pretrained structure prediction models. *Nature Methods*, 21(11):2107–2116, 2024.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Zili Lu, Heng Pan, Yueyue Dai, Xueming Si, and Yan Zhang. Federated learning with non-iid data: A survey. *IEEE Internet of Things Journal*, 11(11):19188–19209, 2024.

Jian-Hao Luo and Jianxin Wu. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791*, 2017.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

Yuzhu Mao, Zihao Zhao, Guangfeng Yan, Yang Liu, Tian Lan, Linqi Song, and Wenbo Ding. Communication-efficient federated learning with adaptive quantization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022.

Adrian Martin, Isabel de-la Bandera, Adriano Mendo, Jose Outes, Juan Ramiro, and Raquel Barco. Federated deep reinforcement learning for endc optimization. *IEEE Transactions on Mobile Computing*, 2025.

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

Fanxu Meng, Hao Cheng, Ke Li, Huixiang Luo, Xiaowei Guo, Guangming Lu, and Xing Sun. Pruning filter in filter. *Advances in Neural Information Processing Systems*, 33:17629–17640, 2020.

Jiaxu Miao, Zongxin Yang, Leilei Fan, and Yi Yang. Fedseg: Class-heterogeneous federated learning for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8042–8052, 2023.

Thomas Minka. Estimating a dirichlet distribution, 2000.

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

Manh Duong Nguyen, Trung Thanh Nguyen, Huy Hieu Pham, Trong Nghia Hoang, Phi Le Nguyen, and Thanh Trung Huynh. Fedmac: Tackling partial-modality missing in federated learning with cross-modal aggregation and contrastive regularization. In *2024 22nd International Symposium on Network Computing and Applications (NCA)*, pages 278–285. IEEE, 2024.

Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE, 2019.

Yves Normandin. Hidden markov models, maximum mutual information estimation, and the speech recognition problem. 1991.

Kenny Olsen, Rasmus M Hoeegh Lindrup, and Morten Mørup. Think global, adapt local: Learning locally adaptive k-nearest neighbor kernel density estimators. In *International Conference on Artificial Intelligence and Statistics*, pages 4114–4122. PMLR, 2024.

Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33:2925–2934, 2020.

Jiaming Pei and Wei Li. Clustered federated learning with slightly skewed labels. In *Proceedings of the ICLR Tiny Paper Track*, 2023.

Jiaming Pei and Wei Li. Dual model pruning enables efficient federated learning in intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

Jiaming Pei and Wei Li. Unveiling the effects of slightly skewed labels on traffic data analysis. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

Jiaming Pei, Wei Li, and Shahid Mumtaz. From routine to reflection: Pruning neural networks in communication-efficient federated learning. *IEEE Transactions on Artificial Intelligence*, 2024.

Yuhuai Peng, Alireza Jolfaei, Qiaozhi Hua, Wen-Long Shang, and Keping Yu. Real-time transmission optimization for edge computing in industrial cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 18(12):9292–9301, 2022.

Md Ferdous Pervej, Richeng Jin, and Huaiyu Dai. Hierarchical federated learning in wireless networks: Pruning tackles bandwidth scarcity and system heterogeneity. *IEEE Transactions on Wireless Communications*, pages 1–1, 2024.

Javier Poyatos, Daniel Molina, Aritz D Martinez, Javier Del Ser, and Francisco Herrera. Evoprunedeeptl: An evolutionary pruning model for transfer learning based deep neural networks. *Neural Networks*, 158:59–82, 2023.

Pavana Prakash, Jiahao Ding, Rui Chen, Xiaoqi Qin, Minglei Shu, Qimei Cui, Yuanxiong Guo, and Miao Pan. Iot device friendly and communication-efficient federated learning via joint model pruning and quantization. *IEEE Internet of Things Journal*, 9(15):13638–13650, 2022.

John Rachwan, Daniel Zügner, Bertrand Charpentier, Simon Geisler, Morgane Ayle, and Stephan Günnemann. Winning the lottery ahead of time: Efficient early network pruning. In *International Conference on Machine Learning*, pages 18293–18309. PMLR, 2023.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018.

Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108, pages 2021–2031. PMLR, 2020.

Abderahman Rejeb, Karim Rejeb, Horst Treiblmaier, Andrea Appolloni, Salem Alghamdi, Yaser Alhasawi, and Mohammad Iranmanesh. The internet of things (iot) in healthcare: Taking stock and moving forward. *Internet of Things*, 22:100721, 2023.

Yuji Roh, Kangwook Lee, Steven Whang, and Changho Suh. Sample selection for fair and robust training. *Advances in Neural Information Processing Systems*, 34:815–827, 2021.

Yichen Ruan and Carlee Joe-Wong. Fedsoft: Soft clustered federated learning with proximal local updating. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8124–8131, 2022.

Batool Salehi, Debashri Roy, Jerry Gu, Chris Dick, and Kaushik Chowdhury. Flash-and-prune: Federated learning for automated selection of high-band mmwave sectors using model pruning. *IEEE Transactions on Mobile Computing*, 2024.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural*

networks and learning systems, 31(9):3400–3413, 2019.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

Haopu Shang, Jia-Liang Wu, Wenjing Hong, and Chao Qian. Neural network pruning by cooperative coevolution. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4814–4820. ijcai.org, 2022.

Donald Shenaj, Eros Fanì, Marco Toldo, Debora Caldarola, Antonio Tavera, Umberto Michieli, Marco Ciccone, Pietro Zanuttigh, and Barbara Caputo. Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 444–454, 2023.

Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

Yong Shi, Anda Tang, Lingfeng Niu, and Ruizhi Zhou. Sparse optimization guided pruning for neural networks. *Neurocomputing*, 574:127280, 2024.

Sietsma and Dow. Neural net pruning-why and how. In *IEEE 1988 international conference on neural networks*, pages 325–333. IEEE, 1988.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1453–1460, 2011.

Wensheng Su, Zhenni Li, Minrui Xu, Jiawen Kang, Dusit Niyato, and Shengli Xie. Compressing deep reinforcement learning networks with a dynamic structured pruning method for autonomous driving. *IEEE Transactions on Vehicular Technology*, 73(12):18017–18030, 2024.

Tao Sun, Dongsheng Li, and Bao Wang. Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4289–4301, 2022.

Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 9953–9961, 2023.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.

Janvi Thakkar and Devvrat Joshi. Fedspectral+: Spectral clustering using federated learning. *arXiv preprint arXiv:2302.02137*, 2023.

Vicenç Torra. A systematic construction of non-iid data sets from a single data set: non-identically distributed data. *Knowledge and Information Systems*, 65(3):991–1003, 2023.

Nguyen H Tran, Wei Bao, Albert Zomaya, Minh NH Nguyen, and Choong Seon Hong. Federated learning over wireless networks: Optimization model design and analysis. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 1387–1395. IEEE, 2019.

Saeed Vahidian, Mahdi Morafah, and Bill Lin. Personalized federated learning by structured and unstructured pruning under data heterogeneity. In *2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 27–34. IEEE, 2021.

Anish K Vallapuram, Pengyuan Zhou, Young D Kwon, Lik Hang Lee, Hengwei Xu, and Pan Hui. Hidenseek: Federated lottery ticket via server-side pruning and sign supermask. *arXiv preprint arXiv:2206.04385*, 2022.

Raffaele Viterbo, Marco Piavanini, Lorenzo Italiano, Mattia Brambilla, Mattia Cerutti, Sanders Batista Felix, Simone Specchia, Edoardo Piantoni, Giovanni Megna, Diego Franceschini, et al. Leveraging smart tunnel systems: V2x-driven positioning for cavs in gnss-denied scenarios. In *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 01–06. IEEE, 2025.

Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A practical guide, 1st ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

Chaoqun Wan, Xu Shen, Yonggang Zhang, Zhiheng Yin, Xinmei Tian, Feng Gao, Jianqiang Huang, and Xian-Sheng Hua. Meta convolutional neural networks for single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4682–4691, 2022.

Haoyu Wang and Wei-Qiang Zhang. Unstructured pruning and low rank factorisation of self-supervised pre-trained speech models. *IEEE Journal of Selected Topics in Signal Processing*, 18(6):1046–1058, 2024.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2019.

Zhiguo Wang, Xintong Wang, Ruoyu Sun, and Tsung-Hui Chang. Federated semi-supervised learning with class distribution mismatch. *arXiv preprint arXiv:2111.00010*, 2021.

Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14913–14922, 2021.

Xi Weng, Yan Yan, Si Chen, Jing-Hao Xue, and Hanzi Wang. Stage-aware feature alignment network for real-time semantic segmentation of street scenes. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(7):4444–4459, 2021.

Jun Wu, Jingrui He, and Elizabeth Ainsworth. Non-iid transfer learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10342–10350, 2023.

Tingting Wu, Chunhe Song, and Peng Zeng. Efficient federated learning on resource-constrained edge devices based on model pruning. *Complex & Intelligent Systems*, 9(6):6999–7013, 2023.

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

Hui Xia, Shuo Xu, Jiaming Pei, Rui Zhang, Zhi Yu, Weitao Zou, Lukun Wang, and Chao Liu. Fedme 2: Memory evaluation & erase promoting federated unlearning in dtmn. *IEEE Journal on Selected Areas in Communications*, 2023.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yuxi Xie, Min Qiu, Haibo Zhang, Lizhi Peng, and Zhenxiang Chen. Gaussian distribution based oversampling for imbalanced data classification. *IEEE Transactions on Knowledge and Data Engineering*, 34(2):667–679, 2020.

Yuanhao Xiong, Ruochen Wang, Minhao Cheng, Felix Yu, and Cho-Jui Hsieh. Feddm: Iterative distribution matching for communication-efficient federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16323–16332, 2023.

Ao Xiong, Han Zhou, Yu Song, Dong Wang, Xu Wei, Da Li, and Bo Gao. A multi-task based clustering personalized federated learning method. *Big Data Mining and Analytics*, 7(4):1017–1030, 2024.

Li Da Xu and Lian Duan. Big data for cyber physical systems in industry 4.0: a survey. *Enterprise Information Systems*, 13(2):148–169, 2019.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Dongkuan Xu, Ian EH Yen, Jinxi Zhao, and Zhibin Xiao. Rethinking network pruning–under the pre-train and fine-tune paradigm. *arXiv preprint arXiv:2104.08682*, 2021.

Wenyuan Xu, Weiwei Fang, Yi Ding, Meixia Zou, and Naixue Xiong. Accelerating federated learning for iot in big data analytics with pruning, quantization and selective updating. *IEEE Access*, 9:38457–38466, 2021.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

Liping Yi, Gang Wang, Xiaoguang Liu, Zhuan Shi, and Han Yu. Fedgh: Heterogeneous federated learning with generalized global header. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 8686–8696, 2023.

Liping Yi, Xiaorong Shi, Nan Wang, Jinsong Zhang, Gang Wang, and Xiaoguang Liu. Fedpe: Adaptive model pruning-expanding for federated learning on mobile devices. *IEEE Transactions on Mobile Computing*, 2024.

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Xin Yu, Thiago Serra, Srikumar Ramalingam, and Shandian Zhe. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks. In *International Conference on Machine Learning*, pages 25668–25683. PMLR, 2022.

Yang Yu, Qi Liu, Likang Wu, Runlong Yu, Sanshi Lei Yu, and Zaixi Zhang. Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4854–4863, 2023.

Dun Zeng, Siqi Liang, Xiangjing Hu, Hui Wang, and Zenglin Xu. Fedlab: A flexible federated learning framework. *Journal of Machine Learning Research*, 24(100):1–7, 2023.

Yuyao Zhang and Nikolaos M Freris. Adaptive filter pruning via sensitivity feedback. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):10996–11008, 2023.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Hong Zhang, Ji Liu, Juncheng Jia, Yang Zhou, Huaiyu Dai, and Dejing Dou. Fedduap: Federated learning with dynamic update and adaptive pruning using shared data on the server. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2776–2782. ijcai.org, 2022.

Yushun Zhang, Xing Zhang, and Yizhuo Cai. Multi-task federated learning based on client scheduling in mobile edge computing. In *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 185–190. IEEE, 2022.

Xiao Zhang, Qilin Wang, Ziming Ye, Haochao Ying, and Dongxiao Yu. Federated representation learning with data heterogeneity for human mobility prediction. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

Rongqing Zhang, Jingxin Mao, Hanqiu Wang, Bing Li, Xiang Cheng, and Liuqing Yang. A survey on federated learning in intelligent transportation systems. *IEEE Transactions on Intelligent Vehicles*, 2024.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Hailiang Zhao, Xueyan Tang, Peng Chen, Jianwei Yin, and Shuiguang Deng. Data-locality-aware task assignment and scheduling for distributed job executions. *IEEE Transactions on Services Computing*, 2025.

Jiali Zheng and Jing Tang. Communication-efficient federated learning based on compressed sensing and ternary quantization. *Applied Intelligence*, 55(2):100, 2025.

Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

Hongliang Zhou, Yifeng Zheng, Hejiao Huang, Jiangang Shu, and Xiaohua Jia. Toward robust hierarchical federated learning in internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

A Appendix A

This appendix provides the complete derivations of Theorem 4.1. We first state the auxiliary results (Theorem A1, Lemma A1, Lemma A2) that are used in the proof.

Before the proof, some necessary assumptions are stated as following:

ASSUMPTION 1. *$F(.)$ is $\beta - smooth$: for all a and b, $\| \nabla F(a) - \nabla F(b) \| \leq \beta \| a - b \|$, for all $k = 1, ..., K$.*

ASSUMPTION 2. *F_k is $\rho - Lipschitz$: for all a and b, $\| F_k(a) - F_k(b) \| \leq \rho \| a - b \|$, $\forall a, b \in \mathbb{R}^d, \rho > 0$, for all $k = 1, ..., K$.*

ASSUMPTION 3. *The gradient ∇F_k is upper bounded, i.e., $\mathbb{E} \| \nabla F_k(w_t) \|^2 < G^2$, for all $k = 1, 2, .., K$ and $t = 0, ..., T + 1$.*

THEOREM A1. *For any t, we have*

$$\| w^{ave}(t) - w^g(t) \|^2 \leq 4N(1 + \eta\beta)^2\eta^2(E - 1)^2G^2. \tag{A.1}$$

PROOF. We first bound the deviation between the local updates $w'_n(t-1)$ and $w_n(t-1)$.

$$\|w'_n(t-1) - w_n(t-1)\|^2 \leq 4\eta^2(E-1)^2G^2, \tag{A.2}$$

where the bound follows from Assumption 3 (bounded gradients).

Now, consider the difference between the global and averaged models:

$$\|w^g(t) - w^{ave}(t)\| = \left\| \sum_{n=1}^{N} \frac{D^n}{D} \Big((w_n^c)'(t-1) - w_n^c(t-1) \Big) \right\| \tag{A.3}$$

$$\leq \sum_{n=1}^{N} \frac{D^n}{D} \|(w_n^c)'(t-1) - w_n^c(t-1)\|. \tag{A.4}$$

By the update rule of gradient descent, we have

$$(w_n^c)'(t-1) - w_n^c(t-1) = \big((w_n^c)'(t-1) - w_n^c(t-1) \big) \tag{A.5}$$

$$- \eta \Big(\nabla F((w_n^c)'(t-1)) - \nabla F(w_n^c(t-1)) \Big). \tag{A.6}$$

Applying the triangle inequality and the β-smoothness of $F(\cdot)$ (Assumption 1), we get

$$\|(w_n^c)'(t-1) - w_n^c(t-1)\| \leq (1+\eta\beta)\|(w_n^c)'(t-1) - w_n^c(t-1)\|. \quad \text{(by Assumption 1)}$$

Therefore,

$$\|w^g(t) - w^{ave}(t)\| \leq (1+\eta\beta)\sum_{n=1}^{N} \frac{D^n}{D} \|(w_n^c)'(t-1) - w_n^c(t-1)\|. \tag{A.7}$$

Next, taking expectation gives

$$\mathbb{E}\sum_{n=1}^{N} \tfrac{D^n}{D} \|(w_n^c)'(t-1) - w_n^c(t-1)\|^2 \tag{A.8}$$

$$\leq 2\,\mathbb{E}\sum_{n=1}^{N} \tfrac{D^n}{D} \|(w_n^c)'(t-1) - w_n^c(t_0)\|^2 \tag{A.9}$$

$$+ 2\,\mathbb{E}\sum_{n=1}^{N} \tfrac{D^n}{D} \|w_n^c(t-1) - w_n^c(t_0)\|^2. \tag{A.10}$$

Since $t - t_0 < E - 1$, applying Jensen's inequality yields

$$\|(w_n^{c'}(t-1) - w_n^c(t_0))\|^2 \leq (t-t_0)\sum_{\tau=t_0}^{t-2} \eta_\tau^2 \|\nabla F(w_n^c(\tau))\|^2. \quad \text{(by Assumption 3)}$$

Using $\mathbb{E}\|\nabla F_n(w_t)\|^2 < G^2$ from Assumption 3, we obtain

$$\mathbb{E}\sum_{n=1}^{N} \tfrac{D^n}{D} \|(w_n^c)'(t-1) - w_n^c(t-1)\|^2 \leq 4\,\eta^2(E-1)^2G^2. \tag{A.11}$$

Finally, combining the above inequality with the smoothness factor $(1+\eta\beta)^2$ (Assumption 1), we arrive at

$$\|w^{ave}(t) - w^g(t)\|^2 \leq 4N(1+\eta\beta)^2\eta^2(E-1)^2G^2. \tag{A.12}$$

This completes the proof. □

LEMMA A1. *For any t, when $\eta \leq \frac{1}{\beta}$, we have the following inequality*

$$\begin{aligned} & F(w^{ave}(t) - F(w^{ave}(t-1)) \\ & \leq -\eta_{t-1}(1 - \frac{\beta\eta}{2}) \parallel \nabla F(w^{ave}(t-1)) \parallel^2 \end{aligned} \tag{A.13}$$

PROOF. Because $F(.)$ is $\beta - smooth$, we have

$$\begin{aligned} & F(w^{ave}(t) - F(w^{ave}(t-1)) \\ & \leq \nabla F(w^{ave}(t-1))^T (w^{ave}(t) - w^{ave}(t-1)) \\ & + \frac{\beta}{2} \parallel w^{ave}(t) - w^{ave}(t-1) \parallel^2 \\ & \leq -\eta_{t-1}(1 - \frac{\beta\eta}{2}) \parallel \nabla F(w^{ave}(t-1)) \parallel^2 \end{aligned} \tag{A.14}$$

□

DEFINITION 1. *In a given t-th round, we define $\theta(t) = F(w^{ave}(t)) - F(w^*)$.*

LEMMA A2. *According to **Definition 1** and **Lemma A1**, we have*

$$\frac{1}{\theta(T)} - \frac{1}{\theta(0)} \geq T\sigma\eta(1 - \frac{\beta\eta}{2}) \tag{A.15}$$

where $\sigma = min_t \frac{1}{\|w^{ave}(t) - w^*\|^2}$

PROOF. From **Lemma A1**, we have

$$\begin{aligned} & \theta(t) - \theta(t-1) \\ & = F(w^{ave}(t) - F(w^*) - (F(w^{ave}(t-1)) - F(w^*) \\ & = F(w^{ave}(t) - F(w^{ave}(t-1) \\ & \leq -\eta_{t-1}(1 - \frac{\beta\eta}{2}) \parallel \nabla F(w^{ave}(t-1)) \parallel^2 \end{aligned} \tag{A.16}$$

Another form of the above inequality is

$$\theta(t) \leq \theta(t-1) - \eta_{t-1}(1 - \frac{\beta\eta}{2}) \parallel \nabla F(w^{ave}(t-1)) \parallel^2 \tag{A.17}$$

Assume $F(.)$ is convex and using the Cauchy-Schwarz inequality Bhatia and Davis (1995), we have

$$\begin{aligned}\theta(t-1) &= F(w^{ave}(t-1)) - F(w^*) \\ &\leq \nabla F(w^{ave}(t-1))^T (w^{ave}(t-1) - w^*) \\ &\leq \| \nabla F(w^{ave}(t-1)) \| \| w^{ave}(t-1) - w^* \|\end{aligned} \tag{A.18}$$

Substituting (A.18) into (A.17), we get

$$\begin{aligned}\theta(t) &\leq \theta(t-1) - \frac{\eta_{t-1}(1-\frac{\beta\eta}{2})\theta(t-1)^2}{\| w^{ave}(t) - w^* \|^2} \\ &\leq \theta(t-1) - \sigma\eta_{t-1}(1-\frac{\beta\eta}{2})\theta(t-1)^2\end{aligned} \tag{A.19}$$

As $\theta(t)\theta(t-1) > 0$, we have

$$\frac{1}{\theta(t)} - \frac{1}{\theta(t-1)} \geq \sigma\eta_{t-1}(1-\frac{\beta\eta}{2}) \tag{A.20}$$

When $t = T$, we have

$$\sum_{t=1}^{T} \frac{1}{\theta(t)} - \frac{1}{\theta(t-1)} \geq T\sigma\eta(1-\frac{\beta\eta}{2}) \tag{A.21}$$

□

THEOREM 4.1. *For any $\eta \leq \frac{1}{\beta}$, the upper bound of $F(w^g(T)) - F(w^*)$ after T iterations is*

$$F(w^g(T)) - F(w^*) \leq \frac{\xi^2}{2N\rho(1+\eta\beta)(E-1)G + T\sigma(1-\frac{\beta\eta}{2})\xi^2}. \tag{4.17}$$

PROOF. We define $F(w^{ave}(t)) - F(w^*) \geq \xi$, then we have

$$\begin{aligned}&\frac{1}{F(w^g(T)) - F(w^*)} - \frac{1}{\theta(T)} \\ &= \frac{\theta(T) - (F(w^g(T)) - F(w^*))}{(F(w^g(T)) - F(w^*))\theta(T)}\end{aligned} \tag{A.22}$$

According to **Definition 1**, we get

$$\begin{aligned}&\frac{\theta(T)-(F(w^g(T))-F(w^*))}{(F(w^g(T))-F(w^*))\theta(T)}\\&=\frac{F(w^{ave}(T))-F(w^g(T))}{(F(w^g(T))-F(w^*))\theta(T)}\\&\geq\frac{2\rho N(1+\eta\beta)\eta(E-1)G}{\xi^2}\end{aligned}\tag{A.23}$$

Summing up (A.15) and (A.22), we get

$$\begin{aligned}&\frac{1}{F(w^g(T))-F(w^*)}\\&\geq\frac{1}{F(w^g(T))-F(w^*)}-\frac{1}{\theta(0)}\\&\geq\frac{2\rho N(1+\eta\beta)\eta(E-1)G+T\sigma\eta(1-\frac{\beta\eta}{2})}{\xi^2}\end{aligned}\tag{A.24}$$

Taking the reciprocal of the above inequality yields

$$\begin{aligned}&F(w^g(T))-F(w^*)\leq\\&\frac{\xi^2}{2N\rho(1+\eta\beta)(E-1)G+T\sigma(1-\frac{\beta\eta}{2})\xi^2}\end{aligned}\tag{A.25}$$

□

COROLLARY 4.1. *Under the conditions of Theorem 4.1, choose constants* $c,\alpha\in(0,1]$ *and set the stepsize and local period as*

$$\eta\;=\;\frac{c\,N}{\sqrt{T}},\qquad E\;=\;\left\lfloor\frac{\alpha\,T}{N^2}\right\rfloor,$$

with $\eta\leq 1/\beta$ *and* T *large enough so that* $E\geq 1$. *Then the global model satisfies*

$$F(w^g(T))-F(w^*)\;\leq\;\frac{\xi^2}{c\alpha\rho G\,\sqrt{T}+\frac{c\sigma}{2}\,N\sqrt{T}}\;\leq\;\frac{2\xi^2}{c\sigma}\cdot\frac{1}{N\sqrt{T}}\tag{4.18}$$

PROOF. By Theorem 4.1, we have

$$F(w^g(T))-F(w^*)\;\leq\;\frac{\xi^2}{2N\rho(1+\eta\beta)\,\eta\,(E-1)G\;+\;T\sigma\,\eta\Big(1-\frac{\beta\eta}{2}\Big)}.\tag{A.26}$$

Choose constants $c, \alpha \in (0, 1]$ and set $\eta = c\,N/\sqrt{T}$ and $E = \lfloor \alpha T/N^2 \rfloor$. Assume T is large enough that $E \geq 1$, $E - 1 \geq \frac{\alpha}{2}\frac{T}{N^2}$, and $\beta\eta \leq \frac{1}{2}$ (e.g., $T \geq \max\{2N^2/\alpha,\, (2\beta cN)^2\}$). Then $1 + \eta\beta \leq \frac{3}{2}$ and $1 - \frac{\beta\eta}{2} \geq \frac{3}{4}$, so the denominator of $(A.26)$ is bounded below by

$$\begin{aligned} &2N\rho(1+\eta\beta)\,\eta\,(E-1)G + T\sigma\,\eta\Big(1 - \tfrac{\beta\eta}{2}\Big) \\ &\geq c\,\alpha\,\rho G\,\frac{T}{N}\cdot\frac{N}{\sqrt{T}} \;+\; \frac{c\sigma}{2}\,T\cdot\frac{N}{\sqrt{T}} = c\alpha\rho G\,\sqrt{T} \;+\; \frac{c\sigma}{2}\,N\sqrt{T}. \end{aligned} \tag{A.27}$$

Hence,

$$F(w^g(T)) - F(w^*) \;\leq\; \frac{\xi^2}{c\alpha\rho G\,\sqrt{T} + \frac{c\sigma}{2}\,N\sqrt{T}} \;\leq\; \frac{2\xi^2}{c\sigma}\cdot\frac{1}{N\sqrt{T}} \;=\; \mathcal{O}\Big(\frac{1}{N\sqrt{T}}\Big). \tag{A.28}$$

□

From **Corollary 4.1**, our method has a convergence rate of $\mathcal{O}(\frac{1}{N\sqrt{T}})$, which is the fastest convergence in cross-device FL, as shown in Table 6.2. Moreover, our method has a communication complexity of $\mathcal{O}(TN^2)$.

B Appendix B

In this appendix, we provide the detailed proofs of the Theorems stated in Section 6.4. These proofs establish the convergence of the Gaussian Mixture Oversampling (GMO) method, the stability of mixture coefficients under non-i.i.d. data distributions, and the convergence of the Gaussian Mixture Clustering for Federated Learning (GMCFL) method.

THEOREM 6.1. ***Convergence of GMO*** *According to Equation (6.6), the approximate objective function obtained by sampling with Gaussian mixture sampling in FL can converge to the expected true objective function,* $F(w) \approx \mathbb{E}_{x\ q}[L(w;x)]$.

PROOF. For each Gaussian mixture component j, we have the probability $q(x) = \sum_{j=1}^{M_i} \alpha_j \cdot \mathcal{N}(x;\mu_j,C_j)$. According to the objective function of Federated learning, we have

$$F(w_n^c) \approx \frac{1}{m_n} \sum_{x \in D_n'} L(w;x) \tag{A.29}$$

where m_n is the number of the synthetic samples. In federated learning, the approximate values of $F(w_n^c)$ can be averaged with weights and we can obtain:

$$F(w) = \sum_{n=1}^{N} \frac{D_n}{D} F(w_n^c) \approx \sum_{n=1}^{N} \sum_{x \in D_n'} \frac{1}{mD_n} L(w;x) = \frac{1}{num} \sum_{n=1}^{N} \sum_{x \in D_n'} \frac{num}{mD_n} L(w;x) \tag{A.30}$$

where $num = \sum_{n=1}^{N} D_n$. According to the Central Limit Theorem, when the sample size is large enough, the sample mean follows a normal distribution. Therefore, $F(w)$ can be approximated as: $F(w) \approx \mathbb{E}_{x\ q}[L(w;x)]$. □

THEOREM 6.2. ***Stability of mixture coefficients*** *Due to fluctuations and shifts in the local data distribution, the estimation of the weight and mean for each Gaussian distribution may contain errors, subsequently impacting the performance of the global model. The expected value of* ϵ_n *represents the upper bound of the estimation error in the algorithm, from which*

we can derive

$$(\mathbb{E}[\epsilon_n])^2 \leq \mathbb{E}\left[\max_j \frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\frac{p_n(x_{ni})}{\sum_{j=1}^{J} \hat{p}_j \hat{\alpha}_j(x_{ni})} - \frac{\hat{\alpha}_{j'}}{\sum_{j=1}^{J} \hat{p}_j \hat{\alpha}_j(x_{ni})}(1 - D_{KL}(p_j||p_{j'}))\right)^2\right]$$
$$+ \max_{j,j'} \left\{\mathbb{E}\left[\frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\hat{\alpha}_j \hat{p}_j(x_{ni}) - \alpha_j \frac{p_j(x_{ni})}{p_{j'}(x_{ni})}\right)^2\right]\right\}$$
$$+ \mathbb{E}\left[\frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\hat{\alpha}_{j'} \hat{p}_j(x_{ni}) D_{KL}(p_j||p_{j'}) + \frac{\hat{\alpha_{j'}} p_j(x_{ni})}{p_{j'}(x_{ni})} + \frac{\hat{\alpha_{j'}} p_{j'}(x_{ni})}{p_j(x_{ni})} + \alpha_{j'}\right)\right]$$

PROOF. Let $\hat{p}_n(x)$ donate the data distribution on client n, α_j represents the weight of Gaussian component j. The mixture coefficient estimation error can be described as follows:

$$\epsilon_n = max_k \left| \frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} p_n(x_{ni}) - \hat{\alpha_n} \sum_{i=1}^{Num_n} \sum_{j=1}^{J} \hat{W}_j \hat{p}_j(x_{nj}) \right| \tag{A.31}$$

Difference of data distribution on different clients can be measured by **KL** divergence as follows:

$$D_{KL}(p_n||p_{n'}) = \int p_n log \frac{p_n(x)}{p_{n'}(x)} dx. \tag{A.32}$$

Then put the above formula into the $\max$ function and the derivation is shown in appendix.

Since **KL** divergence is non-negative, we have:

$$\sum_{n=1}^{N} D_{KL}(p_n||p'_n) = 0. \tag{A.33}$$

Put the above equation into $\max$, we can get:

$$\hat{\alpha}_n = \max_j \{\hat{\alpha}_j \exp\{D_{KL}(p_j||p_n)\}\} \tag{A.34}$$

According to the sample number law, when $num_n \to \infty$, the expected value of the mixture coefficient estimation error ϵ_n can be expressed as:

$$\mathbb{E}[\epsilon_n] = \mathbb{E}\left[max_j \left| \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} p_n(x_{nj}) - \hat{\alpha_n} \sum_{i=1}^{Num_n} \sum_{j=1}^{J} \hat{\alpha}_j \hat{p}_j(x_{nj}) \right|\right] \tag{A.35}$$

Substitute equation (A.34) into (A.35) as follows:

$$\mathbb{E}[\epsilon_n] = \mathbb{E}\left[\max_j \left| \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} p_n(x_{nj}) - \max_j \left\{ \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \sum_{j=1}^{J} \hat{\alpha}_{j'} \hat{p}_j(x_{nj}) \exp\{D_{KL}(p_j \| p_{j'})\} \right\} \right| \right]$$
$$\leq \mathbb{E}\left[\max_j \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \frac{p_n(x_{nj})}{\sum_{j=1}^{J} \hat{\alpha}_{j'} \hat{p}_j(x_{nj}) \exp\{D_{KL}(p_j \| p_{j'})\}} - \frac{\hat{\alpha}_{j'}}{\sum_{j=1}^{J} \hat{\alpha}_{j'} \hat{p}_j(x_{nj}) \exp\{D_{KL}(p_j \| p_{j'})\}} \right| \right]$$
$$+ \mathbb{E}\left[\max_j \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \sum_{j=1}^{J} \hat{\alpha}_{j'} \hat{p}_j(x_{nj}) \exp\{D_{KL}(p_j \| p_{j'})\} - \sum_{j=1}^{J} \alpha_{j'} \frac{p_j(x_{nj})}{p_{j'}(x_{nj})} \right| \right]. \tag{A.36}$$

According to Jensen inequality, we get the lower bound of the ϵ_n as follows:

$$exp\{D_{KL}(p_j||p_{j'}) \geq 1 - D_{KL}(p_j||p_{j'})\} \tag{A.37}$$

Substitute it into equation (A.36), we get

$$\mathbb{E}[\epsilon_n] \leq \mathbb{E}\left[max_j \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \frac{p_n(x_{nj})}{\sum_{j=1}^{J} \hat{\alpha_{j'}} \hat{p}_j(x_{nj})} - \frac{\hat{\alpha_{j'}}}{\sum_{j=1}^{J} \hat{\alpha_{j'}} \hat{p}_j(x_{nj})} (1 - D_{KL}(p_j||p_{j'})) \right| \right]$$
$$+ \mathbb{E}\left[max_j \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \sum_{j=1}^{J} \hat{\alpha_{j'}} \hat{p}_j(x_{nj})(1 - D_{KL}(p_j||p_{j'})) - \sum_{j=1}^{J} \alpha_{j'} \frac{p_j(x_{nj})}{p_{j'}(x_{nj})} \right| \right] \tag{A.38}$$

For the first inequality, we can derive according to Cauchy-Schwarz inequality as follows:

$$\mathbb{E}[\epsilon_n] \leq \mathbb{E}\left[max_j \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \frac{p_n(x_{nj})}{\sum_{j=1}^{J} \hat{\alpha_{j'}} \hat{p}_j(x_{nj})} - \frac{\hat{\alpha_{j'}}}{\sum_{j=1}^{J} \hat{\alpha_{j'}} \hat{p}_j(x_{nj})} (1 - D_{KL}(p_j||p_{j'})) \right| \right]$$
$$\leq \sqrt{\mathbb{E}\left[\frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left(\frac{p_n(x_{nj})}{\sum_{j=1}^{J} \hat{\alpha_{j'}} \hat{p}_j(x_{nj})} - \frac{\hat{\alpha_{j'}}}{\sum_{j=1}^{J} \hat{\alpha_{j'}} \hat{p}_j(x_{nj})} (1 - D_{KL}(p_j||p_{j'})) \right)^2 \right]} \tag{A.39}$$

For the second inequality, the max function can be split into two parts according to the inequality $|\max a_1, \ldots, a_n - \max b_1, \ldots, b_n| \leq \max |a_1 - b_1|, \ldots, |a_n - b_n|$ as follows:

$$\begin{aligned}
&\mathbb{E}\left[max_j \frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \sum_{j=1}^{J} \hat{\alpha_{j'}} \hat{p_j}(x_{nj})(1 - D_{KL}(p_j||p_{j'})) - \sum_{j=1}^{J} \alpha_{j'} \frac{p_j(x_{nj})}{p_{j'}(x_{nj})} \right| \right] \\
&\leq max_{j,j'} \left\{ \mathbb{E}\left[\frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \hat{\alpha_{j'}} \hat{p_j}(x_{nj})(1 - D_{KL}(p_j||p_{j'})) - \alpha_{j'} \frac{p_j(x_{nj})}{p_{j'}(x_{nj})} \right| \right] \right\} \\
&\leq max_{j,j'} \left\{ \mathbb{E}\left[\frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \hat{\alpha_{j'}} \hat{p_j}(x_{nj}) - \alpha_{j'} \frac{p_j(x_{nj})}{p_{j'}(x_{nj})} \right| \right] \right\} \\
&+ max_{j,j'} \mathbb{E}\left\{ \left[\frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \hat{\alpha_{j'}} \hat{p_j}(x_{nj}) D_{KL}(p_j||p_{j'}) - \frac{\alpha_{j'} p_j(x_{nj})}{p_{j'}(x_{nj})} + \frac{\alpha_{j'} p_{j'}(x_{nj})}{p_j(x_{nj})} - \alpha_{j'} \right| \right] \right\}
\end{aligned} \tag{A.40}$$

The upper bound of the first inequality with Jensen can be derived as :

$$\mathbb{E}\left[\frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \hat{\alpha_{j'}} \hat{p_j}(x_{nj}) - \alpha_{j'} \frac{p_j(x_{nj})}{p_{j'}(x_{nj})} \right| \right] \leq \sqrt{\mathbb{E}\left[\frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left(\hat{\alpha_{j'}} \hat{p_j}(x_{nj}) - \alpha_{j'} \frac{p_j(x_{nj})}{p_{j'}(x_{nj})} \right)^2 \right]} \tag{A.41}$$

And the second inequality with absolute value inequality is as:

$$\begin{aligned}
&\mathbb{E}\left\{ \left[\frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left| \hat{\alpha_{j'}} \hat{p_j}(x_{nj}) D_{KL}(p_j||p_{j'}) - \frac{\alpha_{j'} p_j(x_{nj})}{p_{j'}(x_{nj})} + \frac{\alpha_{j'} p_{j'}(x_{nj})}{p_j(x_{nj})} - \alpha_{j'} \right| \right] \right\} \\
&\leq \mathbb{E}\left\{ \left[\frac{Num_{nj}}{Num_n} \sum_{i=1}^{Num_n} \left(\hat{\alpha_{j'}} \hat{p_j}(x_{nj}) D_{KL}(p_j||p_{j'}) + \frac{\alpha_{j'} p_j(x_{nj})}{p_{j'}(x_{nj})} + \frac{\alpha_{j'} p_{j'}(x_{nj})}{p_j(x_{nj})} + \alpha_{j'} \right| \right) \right\}
\end{aligned} \tag{A.42}$$

To sum up, substituting the above derived formula into $\mathbb{E}[\epsilon_n]$, we get

$$\begin{aligned}
(\mathbb{E}[\epsilon_n])^2 &\leq \mathbb{E}\left[max_j \frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\frac{p_n(x_{ni})}{\sum_{j=1}^{J} \hat{p_j} \hat{\alpha_j}(x_{ni})} - \frac{\hat{\alpha}_{j'}}{\sum_{j=1}^{J} \hat{p_j} \hat{\alpha_j}(x_{ni})} (1 - D_{KL}(p_j||p_{j'})) \right)^2 \right] \\
&+ max_{j,j'} \left\{ \mathbb{E}\left[\frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\hat{\alpha}_j \hat{p_j}(x_{ni}) - \alpha_j \frac{p_j(x_{ni})}{p_{j'}(x_{ni})} \right)^2 \right] \right\} \\
&+ \mathbb{E}\left[\frac{Num_{jn}}{Num_n} \sum_{i=1}^{Num_n} \left(\hat{\alpha}_{j'} \hat{p_j}(x_{ni}) D_{KL}(p_j||p_{j'}) + \frac{\hat{\alpha_{j'}} p_j(x_{ni})}{p_{j'}(x_{ni})} + \frac{\hat{\alpha_{j'}} p_{j'}(x_{ni})}{p_j(x_{ni})} + \alpha_{j'} \right) \right]
\end{aligned} \tag{A.43}$$

Therefore, a proof of the stability of the Gaussian mixture sampling algorithm in the presence of changes and shifts in the data distribution can be derived. □

THEOREM 6.3. ***Convergence of GMCFL*** *In the FL setting, the Gaussian Mixture clustering algorithm converges as*

$$T \geq \frac{1}{2}\left(\frac{L^2}{\epsilon^2} log \frac{2}{\delta} + \frac{log(\frac{2}{\delta})}{\epsilon^2} log\left(\frac{1}{\theta}\sum_{t=1}^{T} exp\left(\frac{\epsilon^2 num}{L^2 T}\cdot t\right)\right)\right) \tag{6.12}$$

when $num \geq \frac{16L^2 log(\frac{2}{\delta})}{\epsilon^2}$. *(The first item is the relationship between the number of samples and the accuracy, and the second item is the theoretical convergence speed.)*

PROOF. Let the objective function be $f(\Theta^t)$, with Θ^* as the global optimal solution, and the objective function of Stochastic Approximation EM algorithm is as follows:

$$f_t(\Theta) = \frac{1}{Num}\sum_{num=1}^{Num} log\left(\sum_{j=1}^{J}\alpha_j \cdot \mathcal{N}(x|\mu_{ij}, C_{ij})\right) \tag{A.44}$$

The S-EM algorithm uses a small batch of random prototypes instead of global data to simulate the EM algorithm. Assuming that the number of prototypes used in each round of simulation is n, the sample set used in the t round is $D_t = x_1^{(t)}, x_2^{(t)}, \cdots, x_{num}^{(t)}$. Then the random approximation of the target function of the t round:

$$\hat{f_\Theta} = \frac{1}{num}\sum_{n=1}^{num} log\left(\sum_{j=1}^{J}\alpha_j \cdot \mathcal{N}(x_i^t|\mu_{ij}, C_{ij})\right). \tag{A.45}$$

According to the theory of stochastic approximation algorithm, when the sample size n is large enough, the S-EM algorithm can obtain the (ϵ, δ)- Approximate solution, namely:

$$P(f_T(\Theta) \leq f(\Theta^*) + \epsilon) \geq 1 - \delta. \tag{A.46}$$

According to Hoeffding inequality, We have:

$$P\left(|\hat{f}_t(\Theta) - f_t(\Theta)| > \epsilon_T\right) \leq 2exp\left(-\frac{2\epsilon_T^2 n}{L^2}\right), \tag{A.47}$$

where $\epsilon_T = \sqrt{\frac{2L^2 \log(2/\delta)}{numT}}$.

$$P\left(|\hat{f}_t(\Theta) - f_t(\Theta)| > \epsilon_T\right) \leq \frac{Var(\hat{f}_t(\Theta))}{\epsilon_T^2}. \tag{A.48}$$

From equation (A.47 and (A.48), we obatain:

$$Var(\hat{f}_t(\Theta)) \geq \epsilon_T^2 \cdot 2exp\left(-\frac{2\epsilon_T^2 n}{L^2}\right) \tag{A.49}$$

Substituting ϵ_T into the above formula, we have:

$$T \geq \frac{1}{2}\left(\frac{L^2}{\epsilon^2}log\frac{2}{\delta} + \frac{log(\frac{2}{\delta})}{\epsilon^2}log\left(\frac{1}{\theta}\sum_{t=1}^{T} exp\left(\frac{\epsilon^2 num}{L^2 T}\cdot t\right)\right)\right) \tag{A.50}$$

when $num \geq \frac{16L^2 log(\frac{2}{\delta})}{\epsilon^2}$.

□

C Appendix C

In this appendix, we provide the complete proofs of all lemmas, propositions, theorems, and remarks stated in Section 7.3. While the main text focused on presenting the results and their implications for convergence, communication efficiency, and generalization, here we give the detailed derivations under the stated assumptions.

ASSUMPTION 4. *(Global smoothness) The global objective function $F : \mathbb{R}^d \to \mathbb{R}$ is β-smooth, i.e.,*

$$\|\nabla F(a) - \nabla F(b)\| \leq \beta \|a - b\|, \quad \forall a, b \in \mathbb{R}^d. \tag{A.51}$$

ASSUMPTION 5. *(Local smoothness) Each local objective $F_n : \mathbb{R}^d \to \mathbb{R}$ is L-smooth, i.e.,*

$$\|\nabla F_n(a) - \nabla F_n(b)\| \leq L \|a - b\|, \quad \forall a, b \in \mathbb{R}^d, \ \forall n = 1, \ldots, K. \tag{A.52}$$

ASSUMPTION 6. *(Bounded gradients) The local stochastic gradients are uniformly bounded in expectation:*

$$\mathbb{E}\big[\|\nabla F_n(w_t)\|^2\big] \leq G^2, \quad \forall n = 1, \ldots, K, \ t = 0, \ldots, T. \tag{A.53}$$

ASSUMPTION 7. *(Convexity) The global loss function $F(w)$ is convex.*

ASSUMPTION 8. *(Pruning error model) Let g_t denote the actual descent direction used at iteration t:*

$$g_t = \nabla F(w_t) + e_t, \tag{A.54}$$

where e_t is the error induced by pruning. We assume that the pruning error is bounded in one of the following equivalent forms:

$$\|e_t\| \leq c_1 P(t) \, \|w_t\|, \quad \|e_t\| \leq c_2 P(t) \, \|\nabla F(w_t)\|, \quad \mathbb{E}\|e_t\|^2 \leq c_3 P(t)^2, \tag{A.55}$$

where $P(t) \in [0, 1]$ is the pruning rate at iteration t and $c_1, c_2, c_3 > 0$ are constants.

LEMMA 7.1. ***Lyapunov Function Monotonicity.*** *Consider the Lyapunov function $V(w) = \|w - w^*\|^2$, where w^* is the global optimum. Under standard gradient descent updates, $V(w)$ decreases monotonically across iterations.*

PROOF. Since $w_{t+1} = w_t - \eta_t \nabla F(w_t)$ and $\nabla F(w^*) = 0$, we expand:

$$\|w_{t+1} - w^*\|^2 = \|w_t - w^* - \eta_t(\nabla F(w_t) - \nabla F(w^*))\|^2. \tag{A.56}$$

Expanding gives

$$\|w_{t+1} - w^*\|^2 = \|w_t - w^*\|^2 - 2\eta_t \langle \nabla F(w_t) - \nabla F(w^*),\, w_t - w^* \rangle + \eta_t^2 \|\nabla F(w_t) - \nabla F(w^*)\|^2. \tag{A.57}$$

By cocoercivity of smooth convex functions (from Assumptions 4, 7),

$$\langle \nabla F(w_t) - \nabla F(w^*),\, w_t - w^* \rangle \geq \tfrac{1}{\beta} \|\nabla F(w_t) - \nabla F(w^*)\|^2. \tag{A.58}$$

Therefore,

$$\|w_{t+1} - w^*\|^2 \leq \|w_t - w^*\|^2 - \left(\tfrac{2\eta_t}{\beta} - \eta_t^2\right) \|\nabla F(w_t)\|^2. \tag{A.59}$$

Since $0 < \eta_t < 2/\beta$, the coefficient is positive, implying

$$\|w_{t+1} - w^*\|^2 \leq \|w_t - w^*\|^2. \tag{A.60}$$

Thus $V(w_t) = \|w_t - w^*\|^2$ is monotonically nonincreasing. □

PROPOSITION 7.1. ***Global Loss Function Monotonicity with Pruning.*** *Let $F(w)$ be convex with Lipschitz continuous gradients. If the learning rate is chosen as*

$$\eta_t = \min\left(\tfrac{2(F(w_t) - F(w^*))}{\|\nabla F(w_t)\|^2},\, \tfrac{1}{L}\right),$$

then the global loss $F(w)$ decreases monotonically across iterations, even in the presence of pruning perturbations.

PROOF. This proposition builds directly on Lemma 7.1 (Lyapunov monotonicity) and additionally uses Assumptions 5 (local L-smoothness) and 8 (pruning error model). It shows how monotonic descent of the Lyapunov function translates into monotonic descent of the global loss $F(w)$ even under pruning perturbations.

Let the actual update direction be $g_t = \nabla F(w_t) + e_t$ with pruning error e_t. The update rule is $w_{t+1} = w_t - \eta_t g_t$.

By β-smoothness (Assumption 4),

$$F(w_{t+1}) \leq F(w_t) + \langle \nabla F(w_t), w_{t+1} - w_t \rangle + \frac{\beta}{2}\|w_{t+1} - w_t\|^2. \tag{A.61}$$

Substituting $w_{t+1} - w_t = -\eta_t g_t$ gives

$$F(w_{t+1}) \leq F(w_t) - \eta_t \langle \nabla F(w_t), g_t \rangle + \frac{\beta \eta_t^2}{2}\|g_t\|^2. \tag{A.62}$$

Expanding $g_t = \nabla F(w_t) + e_t$ yields error terms $\langle \nabla F(w_t), e_t \rangle$ and $\|e_t\|^2$. By Assumption 8, these are bounded by $O(P(t))$. With $\eta_t \leq 1/L$, the quadratic term remains controlled, and with $\eta_t = \frac{2(F(w_t)-F(w^*))}{\|\nabla F(w_t)\|^2}$, the negative descent term dominates.

Therefore $F(w_{t+1}) \leq F(w_t)$, proving that the global loss decreases monotonically under pruning-adjusted updates. □

LEMMA 7.2. ***Impact of Pruning on Gradients and Loss.*** *Let $\mathcal{P}$ denote the pruning operator with pruning rate $P(t)$. Then the perturbation on gradients and the loss function is bounded by*

$$\|\nabla F(\mathcal{P}(w)) - \nabla F(w)\| \leq L \cdot P(t) \cdot \|w\|,$$

ensuring that pruning introduces limited and controllable errors.

PROOF. This Lemma is based on Assumptions 5 (Lipschitz continuous gradients) and 8 (pruning error model). It provides the technical foundation used in Proposition 7.1 and Theorem 7.2, where pruning-adjusted descent is analyzed.

Let $\mathcal{P}(w)$ be the pruning operator applied to parameters w, and denote $w' = \mathcal{P}(w)$. From Assumption 8, pruning perturbation is bounded as

$$\|w' - w\| \ \leq \ P(t)\|w\|, \tag{A.63}$$

with $P(t)$ the dynamic pruning rate at iteration t.

By L-smoothness of each F_n (Assumption 5), we have

$$\|\nabla F(w') - \nabla F(w)\| \;\leq\; L\|w' - w\|. \tag{A.64}$$

Substituting the pruning bound gives

$$\|\nabla F(\mathcal{P}(w)) - \nabla F(w)\| \;\leq\; L\,P(t)\|w\|. \tag{A.65}$$

Over T iterations the cumulative perturbation is

$$\sum_{t=1}^{T} \|\nabla F(\mathcal{P}(w_t)) - \nabla F(w_t)\| \;\leq\; L\sum_{t=1}^{T} P(t)\|w_t\|. \tag{A.66}$$

By smoothness again,

$$|F(w') - F(w)| \;\leq\; L\|w' - w\| \cdot \|w\| \;\leq\; L\,P(t)\|w\|^2. \tag{A.67}$$

Accumulated over training this yields

$$\sum_{t=1}^{T} |F(\mathcal{P}(w_t)) - F(w_t)| \;\leq\; L\sum_{t=1}^{T} P(t)\|w_t\|^2. \tag{A.68}$$

Since $P(t)$ is dynamically scheduled (e.g., linear, exponential, or logarithmic), the cumulative perturbation remains controlled. Thus,

$$F(\mathcal{P}(w_t)) \approx F(w_t), \quad \forall t, \tag{A.69}$$

showing that pruning introduces only limited, well-controlled perturbations to both gradients and the global loss. □

THEOREM 7.1. ***Local Convergence under Dynamic Pruning Rates.*** *During local updates at each client, the model parameters converge towards the local optimum. The convergence rate is preserved up to an additive error term explicitly controlled by the pruning rate* $P(t)$.

PROOF. This Theorem builds upon Lemma 7.2, which bounds the pruning-induced gradient perturbations. The local update at client n can be written as

$$w_n^c(t+1) = w_n^c(t) - \eta\big(\nabla F_n(w_n^c(t)) + e_t\big), \tag{A.70}$$

where e_t denotes the pruning error satisfying $\|e_t\| \leq c_2 P(t)\|\nabla F_n(w_n^c(t))\|$.

By β-smoothness of F_n, we have

$$F_n(w_n^c(t+1)) \leq F_n(w_n^c(t)) - \eta\langle \nabla F_n(w_n^c(t)), \nabla F_n(w_n^c(t)) + e_t\rangle + \tfrac{\beta\eta^2}{2}\|\nabla F_n(w_n^c(t)) + e_t\|^2. \tag{A.71}$$

Rearranging terms and applying the error bound from Lemma 7.2 yields

$$F_n(w_n^c(t+1)) \leq F_n(w_n^c(t)) - \tfrac{\eta}{2}\|\nabla F_n(w_n^c(t))\|^2 + O\big(\eta P(t)\|\nabla F_n(w_n^c(t))\|^2\big). \tag{A.72}$$

Summing over $t = 1, \ldots, T$, we obtain

$$F_n(w_n^c(T)) - F_n(w_n^*) \;\leq\; O\big(\tfrac{1}{T}\big) + O\left(\sum_{t=1}^{T} P(t)\right). \tag{A.73}$$

Thus, the local convergence rate $O(1/T)$ is preserved up to an additive error term that is explicitly controlled by the pruning schedule $P(t)$. □

THEOREM 7.2. ***Global Convergence under Dynamic Pruning Rates.*** *During global aggregation, the federated model converges to a neighborhood of the global optimum. The error floor depends on the cumulative effect of the dynamic pruning rate $P(t)$, which remains bounded throughout training.*

PROOF. This proof builds on Theorem 7.1 (local convergence under pruning) and Lemma 7.2 (bounded pruning perturbations), under Assumptions 4–8. Let $\pi_n := n^n/n$. One global round update is

$$w^g(t+1) = w^g(t) - \eta\, g_t, \qquad g_t = \sum_{n=1}^{N} \pi_n\big(\nabla F_n(w_n^c(t)) + e_t^n\big), \tag{A.74}$$

with $\mathbb{E}\|e_t^n\|^2 \le c_3 P(t)^2$.

By β-smoothness (Assumption 4) of F,

$$F(w^g(t+1)) \le F(w^g(t)) - \eta\langle \nabla F(w^g(t)), g_t\rangle + \tfrac{\beta\eta^2}{2}\|g_t\|^2. \tag{A.75}$$

Writing $g_t = \nabla F(w^g(t)) + \zeta_t$ with

$$\zeta_t := \underbrace{\sum_{n=1}^{N} \pi_n\big(\nabla F_n(w_n^c(t)) - \nabla F_n(w^g(t))\big)}_{\delta_t \text{ (client drift)}} + \underbrace{\sum_{n=1}^{N} \pi_n e_t^n}_{\bar{e}_t \text{ (pruning error)}}, \tag{A.76}$$

and taking $\eta \le \frac{1}{2\beta}$, we obtain constants $c_\kappa \in (0,1)$ and $c_C > 0$ such that

$$F(w^g(t+1)) \le F(w^g(t)) - c_\kappa\,\eta\,\|\nabla F(w^g(t))\|^2 + c_C\,\eta\,\|\zeta_t\|^2. \tag{A.77}$$

By L-smoothness of each F_n (Assumption 5) and bounded gradients (Assumption 6),

$$\mathbb{E}\|\delta_t\|^2 \;\le\; L^2 C_d\,\eta^2 E^2 G^2, \tag{A.78}$$

for some constant $C_d > 0$ depending on the number of local steps E. For the pruning error,

$$\mathbb{E}\|\bar{e}_t\|^2 \;\le\; c_3 P(t)^2. \tag{A.79}$$

Hence,

$$\mathbb{E}\|\zeta_t\|^2 \;\le\; 2L^2 C_d\,\eta^2 E^2 G^2 + 2c_3\,P(t)^2. \tag{A.80}$$

Suppose the training horizon T is divided into three stages of lengths $\Delta_{\mathrm{e}} := t_{\mathrm{early}}$, $\Delta_{\mathrm{m}} := t_{\mathrm{late}} - t_{\mathrm{early}}$, $\Delta_{\mathrm{l}} := T - t_{\mathrm{late}}$. We bound the average squared pruning rate $\frac{1}{T}\sum_{t=0}^{T-1} P(t)^2$ in each stage.

Early stage (linear growth):

$$P(t) = P_{\min} + \frac{P_{\mathrm{mid}} - P_{\min}}{\Delta_{\mathrm{e}}}\,t, \quad t \in [0, \Delta_{\mathrm{e}}]. \tag{A.81}$$

The average square is

$$\mathcal{A}_{\mathrm{e}} = P_{\min}^2 + P_{\min}(P_{\mathrm{mid}} - P_{\min}) + \frac{(P_{\mathrm{mid}} - P_{\min})^2}{3}. \tag{A.82}$$

Middle stage (exponential growth):

$$P(t) = P_{\mathrm{mid}} + (P_{\max} - P_{\mathrm{mid}})(1 - e^{-\alpha s}), \quad s := t - t_{\mathrm{early}}. \tag{A.83}$$

Then

$$\mathcal{A}_{\mathrm{m}} \;\le\; 2P_{\mathrm{mid}}^2 + 2(P_{\max} - P_{\mathrm{mid}})^2 \Big[1 - \frac{2(1 - e^{-\alpha\Delta_{\mathrm{m}}})}{\alpha\Delta_{\mathrm{m}}} + \frac{1 - e^{-2\alpha\Delta_{\mathrm{m}}}}{2\alpha\Delta_{\mathrm{m}}}\Big]. \tag{A.84}$$

Late stage (logarithmic decay):

$$P(t) = P_{\max} - (P_{\max} - P_{\mathrm{mid}})\beta \ln\Big(1 + \frac{T - t}{t_{\mathrm{late}}}\Big). \tag{A.85}$$

A crude bound gives

$$\mathcal{A}_{\mathrm{l}} \;\le\; 2P_{\max}^2 + \frac{2(P_{\max} - P_{\mathrm{mid}})^2\beta^2}{3}\Big(\frac{\Delta_{\mathrm{l}}}{t_{\mathrm{late}}}\Big)^2. \tag{A.86}$$

Combining the three stages,

$$\frac{1}{T}\sum_{t=0}^{T-1} P(t)^2 \;\le\; \frac{\Delta_{\mathrm{e}}}{T}\mathcal{A}_{\mathrm{e}} + \frac{\Delta_{\mathrm{m}}}{T}\mathcal{A}_{\mathrm{m}} + \frac{\Delta_{\mathrm{l}}}{T}\mathcal{A}_{\mathrm{l}} =: \overline{P^2}_{\text{3-stage}}. \tag{A.87}$$

Taking expectations in (A.77), using (A.80), and summing over $t = 0, \ldots, T-1$ yields

$$\mathbb{E}[F(w^g(T))] \le F(w^g(0)) - c_\kappa\eta\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(w^g(t))\|^2 + c_C\eta\sum_{t=0}^{T-1}\Big(2L^2C_d\,\eta^2E^2G^2 + 2c_3P(t)^2\Big). \tag{A.88}$$

By convexity (Assumption 7), this leads to

$$\mathbb{E}[F(w^g(T)) - F^*] \;\le\; \frac{C_1}{\eta T} + C_2\eta^2E^2 + C_3\eta\,\overline{P^2}_{\text{3-stage}}. \tag{A.89}$$

As $T \to \infty$ with fixed η, the first term vanishes. The error floor is

$$\mathsf{Floor} = C_2\eta^2E^2 + C_3\eta\,\overline{P^2}_{\text{3-stage}}, \tag{A.90}$$

where $\overline{P^2}_{\text{3-stage}}$ is explicitly controlled by the three-stage pruning schedule through (A.87). Thus the global model converges to a neighborhood of the optimum, with radius determined by both local drift (η^2E^2) and the cumulative pruning effect from the three stages. □

REMARK 7.1. ***On Pruning-rate Schedules.*** *The proposed pruning schedule $P(t)$ is monotonic within each training stage (early, middle, late), ensuring gradual pruning and stable convergence.*

PROOF. We need to verify that the pruning rate $P(t)$ is monotonic within each stage (early, middle, late) and has a well-defined limit as $t \to T$.

Early Stage ($0 \le t < t_{\text{early}}$): Here $P(t)$ grows linearly from $P_{\min}$ to P_{mid},

$$P(t) = P_{\min} + (P_{\text{mid}} - P_{\min}) \cdot \frac{t}{\Delta_{\text{e}}}, \tag{A.91}$$

so that

$$\frac{dP(t)}{dt} = \frac{P_{\text{mid}} - P_{\min}}{\Delta_{\text{e}}} \ \ge 0. \tag{A.92}$$

Thus $P(t)$ is monotone increasing on $[0, t_{\text{early}}]$.

Middle Stage ($t_{\text{early}} \le t < t_{\text{late}}$): Here $P(t)$ rises exponentially towards $P_{\max}$,

$$P(t) = P_{\text{mid}} + (P_{\max} - P_{\text{mid}})\big(1 - e^{-\alpha(t - t_{\text{early}})}\big). \tag{A.93}$$

Differentiation gives

$$\frac{dP(t)}{dt} = (P_{\max} - P_{\text{mid}})\,\alpha e^{-\alpha(t - t_{\text{early}})} \ \ge 0, \tag{A.94}$$

so $P(t)$ is again monotone increasing.

Late Stage ($t_{\text{late}} \le t \le T$): Here $P(t)$ decays logarithmically,

$$P(t) = P_{\max} - (P_{\max} - P_{\text{mid}})\,\beta \ln\left(1 + \frac{T - t}{\tau_{\text{l}}}\right), \tag{A.95}$$

with $\tau_{\text{l}} > 0$. Its derivative is

$$\frac{dP(t)}{dt} = -\frac{(P_{\max} - P_{\text{mid}})\,\beta}{\tau_{\text{l}} + (T - t)} \;<\; 0, \tag{A.96}$$

so $P(t)$ is monotone decreasing on $[t_{\text{late}}, T]$.

Limit as $t \to T$: Because the logarithmic term vanishes at $t = T$, we have $P(T) = P_{\max}$. Thus the pruning rate converges to $P_{\max}$ at the end of training.

□

PROPOSITION 7.3. ***Communication Overhead Reduction.*** *Let p_t denote the proportion of parameters retained after pruning at round t. Then the communication cost per round is reduced by a factor proportional to $1 - p_t$, while the convergence error increases at most by $O(p_t^2)$.*

PROOF. In federated learning, the dominant communication cost per round comes from transmitting model parameters between server and clients. If the unpruned model has d parameters, then in each round the server broadcasts d parameters to N clients and receives d parameters back, for a total overhead

$$C = 2Nd. \tag{A.97}$$

Suppose pruning retains only a fraction p_t of parameters at round t, i.e., the pruned model has $|w'| = p_t d$ parameters. Then the per-round communication becomes

$$C' = 2Np_t d. \tag{A.98}$$

The relative reduction ratio is

$$R = \frac{C - C'}{C} = 1 - p_t. \tag{A.99}$$

Hence the overhead is reduced by a factor proportional to the fraction pruned $(1 - p_t)$. As $p_t \to 0$ (heavy pruning), the ratio $R \to 1$, signifying nearly complete elimination of communication cost.

At the same time, pruning may introduce perturbations to the gradients, but as shown in Lemma 7.2, these perturbations scale at most quadratically with p_t, i.e. $O(p_t^2)$. Thus the reduction in communication overhead dominates, while the additional convergence error remains controlled.

Therefore, pruning reduces communication overhead per round by a factor $1 - p_t$, with only $O(p_t^2)$ impact on convergence. □

THEOREM 7.3. ***Generalization under Non-IID Pruning.*** *By reducing model complexity, pruning improves the generalization ability of the global model. Under Non-IID data distributions, dynamic pruning rates $P(t)$ mitigate gradient variance across clients, thereby preserving generalization performance.*

PROOF. This Theorem follows from two key observations: (i) pruning reduces model complexity, thereby tightening generalization bounds, (ii) dynamic pruning schedules $P(t)$ help control gradient variance under heterogeneous (Non-IID) client data.

Let w denote the original parameters and $w' = \mathcal{P}(w)$ the pruned parameters. Pruning ensures $\|w'\|_0 \ll \|w\|_0$, hence the VC dimension satisfies

$$VC(w') \leq VC(w). \tag{A.100}$$

The generalization error for a model of VC dimension d satisfies the bound

$$\epsilon(d) \;\leq\; \sqrt{\frac{d \log(n/d)}{n}}, \tag{A.101}$$

where n is the number of training samples. Since $VC(w') \leq VC(w)$, we have

$$\epsilon(w') \leq \epsilon(w). \tag{A.102}$$

Thus pruning reduces hypothesis class complexity and directly improves generalization.

In federated learning, data heterogeneity means local client distributions $\{\mathcal{D}_n\}$ differ, so the global distribution is $\mathcal{D} = \sum_{n=1}^{N} \frac{D^n}{n} \mathcal{D}_n$. This induces gradient variance

$$\mathrm{Var}(\nabla F(w)) = \sum_{n=1}^{N} \frac{D^n}{n} \mathrm{Var}(\nabla F_n(w)), \tag{A.103}$$

which harms generalization.

By Lemma 7.2, pruning introduces only bounded perturbations, while removing redundant parameters reduces sensitivity to client-specific biases. Dynamic pruning schedules $P(t)$ further mitigate heterogeneity:

Early stage: small $P(t)$ retains essential shared features.

Mid stage: increasing $P(t)$ removes redundant parameters, reducing local overfitting.

Late stage: stabilized $P(t)$ prevents over-pruning and maintains balanced gradients.

Combining the above, the generalization error after pruning satisfies

$$\epsilon' \ \leq \ \epsilon + C \cdot \mathrm{Var}(\nabla F(w)), \tag{A.104}$$

for some constant C depending on $P(t)$. Because pruning reduces both $VC(w)$ and $\mathrm{Var}(\nabla F(w))$, the pruned global model generalizes at least as well as the unpruned one, and often better, especially under Non-IID distributions.

□