CRAFT-DFL: NAVIGATING THE PERFORMANCE OF DECENTRALIZED FEDERATED LEARNING DEPLOYMENTS

by

Chengyan Jiang

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2024

Table of Contents

List of Tables

List of Figures

Abstract

The widespread adoption of smartphones and smart wearable devices has led to the widespread use of Centralized Federated Learning (CFL) for training powerful machine learning models while preserving data privacy. However, CFL faces limitations due to overreliance on a central server, which impacts latency and system robustness. Decentralized Federated Learning (DFL) is introduced to address these challenges. DFL facilitates direct collaboration among participating devices without relying on a central server. Each device can independently connect with other devices and share model parameters.

This work explores crucial factors influencing the convergence and generalization capacity of DFL models, emphasizing network topologies, non-IID data distribution, and training strategies. We first derive the convergence rate of different DFL model deployment strategies. Then, we comprehensively analyze various network topologies (e.g., linear, ring, star, and mesh) with different degrees of non-IID data and evaluate them over widely adopted machine-learning models (e.g., classical, deep neural networks, and Large Language Models) and real-world datasets. The results reveal that models converge to the optimal one for IID data. However, the convergence rate is inversely proportional to the degree of non-IID data distribution. Our findings will serve as valuable guidelines for designing effective DFL model deployments in practical applications.

List of Abbreviations Used

FL	Federated Learning
CFL	Centralized Federated Learning
DFL	Decentralized Federated Learning
AR	Augmented reality
CAVs	Connected autonomous vehicle
Resnet	Residual Network
LLM	Large Language Model
NLP	Natural Language Processing
SGD	Stochastic Gradient Descent
NON-IID	Non-independent and identically distributed
NC	Non-converge
C_linear	Continues linear
C_ring	Continues ring
A_linear	Aggregate linear
A_ring	Aggregate ring

Acknowledgements

I received a lot of support and help from many people in completing this thesis, and I sincerely thank them for their assistance.

I would like to thank my professor, Dr. Israat Haque. Throughout my academic journey, she has provided careful and nurturing guidance, helping me grow and become a better researcher. Over the course of two years of research and study, I learned how to conduct research and completed this thesis. When I first joined the lab, I had no idea what research was. With her meticulous help, I successfully completed my thesis. I owe this to her patient and careful guidance. She is an excellent teacher, and like a gardener nurturing young saplings, she helps her students grow.

Also, I want to thank my partner, Jiamingfan. With her help, I found my research direction and completed this paper. She extended a helping hand when I needed it most, even assisting me in revising my thesis late at night. Her companionship enabled me to graduate quickly, and for that, I am very grateful.

Chapter 1

Introduction

Modern applications generate vast amounts of data, e.g., from connected autonomous vehicles (CAVs) [24], augmented reality (AR) [1], Internet of things(IoT) [30, 37], and other network elements. These data form the backbone of data-driven network functions and services. Consequently, there is an increasing need for effective methods to harness this information to enhance network functions and services. The ability to process and analyze such extensive datasets is crucial for improving the efficiency, reliability, and functionality of modern networks.

Centralized Federated Learning (CFL) [26] is one such promising method designed to address these challenges. In CFL, a central server aggregates data from numerous distributed devices to train a global model. Specifically, the server first sends an initial set of parameters to user devices, which then begin local training based on these initial parameters. After the training is completed, the devices upload their trained parameters back to the server. The server then aggregates these parameters. This approach offers significant advantages, particularly for resource-constrained devices, by enabling them to benefit from a collectively trained model without the necessity of directly sharing their data. This model training process ensures data privacy and security, as the individual data remains localized on the devices, mitigating the risks associated with data breaches and unauthorized access. The global model, once trained, can be distributed back to the devices, providing them with the enhanced capabilities derived from the aggregated data insights. This methodology not only optimizes the use of computational resources but also accelerates the development and deployment of intelligent network functions and services [30].

However, CFL faces limitations including single points of failure, communication bottlenecks, and potential privacy breaches. To overcome these issues, Decentralized Federated Learning (DFL) facilitates direct collaboration among devices, eliminating the need for a central server and enhancing robustness and privacy [45] [3]. Each

device has the capability to perform both computation and communication. In DFL, each device can function as either a server or a client device. This distinction provides DFL with greater flexibility, making it more advantageous than CFL in more complex application scenarios.

Specifically, DFL has the following characteristics: each device can function as either a server or a client, each device possesses both transmission and computational capabilities, and each device is networked, allowing for the transmission of parameters over the network. This structure endows DFL with high flexibility and scalability, enabling each node to participate in data processing and computation, as well as to share and update model parameters collaboratively. The decentralized approach not only enhances system fault tolerance and robustness but also better protects data privacy by avoiding the security risks associated with centralized data storage. The effectiveness and reliability of Decentralized Federated Learning (DFL) systems hinge upon several critical factors such as network topology, non-IID data distributions, and training strategies [3] [45].

Network topology, which refers to the arrangement of nodes within a network, plays a crucial role in DFL. Common topologies include ring, linear, star, and mesh structures. The star topology is often used in FL due to its server-client architecture [27] [23]. In DFL, the central client can be seen as either a client or a server [3]. The central node coordinates communication, making it suitable for scenarios requiring centralized control. Ring and linear topologies are popular for their bandwidth and latency advantages in high-performance computing and data centers [34] [42] [44]. These topologies distribute the communication load more evenly across nodes, reducing bottlenecks and potentially increasing fault tolerance and scalability. Mesh topology, with its many interconnected nodes, offers high redundancy and robustness, though it comes at the cost of increased complexity in communication management [5] [29].

Non-IID (non-Independent and Identically Distributed) data distributions occur when data across different nodes is not uniform or identically distributed. This means that each node's data can have different distributions, characteristics, and statistical properties. Unlike IID data, where each node's data is drawn from the same distribution, non-IID data presents significant challenges for DFL by complicating model

generalization, training consistency, and overall system performance [49] [47] [23].

Training strategies, which involve methods and protocols for local model updates and global aggregation, also significantly impact the convergence speed and robustness of DFL systems. There are two main training strategies [45]: *continuous* and *aggregation*. The continuous approach involves a device continuing training based on the parameters from the previous device [34] [42] [44], while the aggregation approach entails devices receiving and aggregating model parameters from other devices, then using the aggregated parameters for subsequent training [27] [31] [22]. In the continuous approach, the model parameters are passed sequentially from one device to another, allowing each device to refine the model further. This method can enhance the model's performance by leveraging the sequential nature of updates. In contrast, the aggregation approach involves periodic aggregation of model parameters from multiple devices, which can improve the model's robustness by integrating diverse updates but may require more sophisticated synchronization mechanisms.

1.1 Contribution

The above factors collectively dictate the effectiveness and reliability of DFL systems. Therefore, a thorough analysis is essential to understand how variations in network topology, non-IID data presence, and different training strategies impact DFL performance. While existing studies have addressed aspects of these challenges, notable gaps persist. Sheller et al. [34] compared convergence performance and accuracy across devices using continual training strategies with linear and ring topologies in DFL. Similarly, Chen et al. [5] introduced a novel DFL solution utilizing a mesh topology and aggregate training strategy. However, neither study investigated the impact of non-IID data, a crucial consideration in real-world applications. Furthermore, in [23], the authors focused on CFL's convergence under varying non-IID data conditions using convex local models, leaving unexplored the efficacy for non-convex models. These gaps highlight the need for comprehensive research to bridge the understanding of how DFL systems perform under diverse conditions, including both convex and non-convex model scenarios.

To address this gap, this thesis thoroughly analyzes these factors and provides insights for designing more efficient and reliable Decentralized Federated Learning

(DFL) frameworks, ensuring they can manage complexities in practical deployments. We evaluate various combinations of network topologies and training strategies. To assess the impact of non-IID data distribution on model performance, we preprocess the dataset using a label imbalance method and quantify non-IID using statistical measures of data dissimilarity or divergence for measurements.

For convex models, we rigorously analyze their convergence and generalization capabilities using mathematical proofs and experiments. In contrast, non-convex models present challenges for analytical proofs. Therefore, we conduct extensive experiments to comprehensively assess their performance under varying degrees of non-IID data. These experiments aim to provide a thorough understanding of how non-convex models perform in real-world applications of decentralized federated learning.

Our main contribution involves:

- This work provides a comprehensive analysis of the factors influencing the convergence efficiency and generalization capacity of Distributed Federated Learning (DFL) models, covering both theoretical foundations and practical considerations.

- We rigorously establish the convergence of different network structures in DFL through convex optimization. This theoretical insight provides valuable guidance for researchers aiming to effectively structure their DFL models.

- To analyse the convergence performance of different model configurations in DFL under non-IID data, we employ various setups, including Continuous Linear, Continuous Ring, Aggregate Ring, Aggregate Linear, Aggregate Mesh, and Aggregate Star. The degree of non-IID data is quantified using a label imbalance method.

- To evaluate the impact of different factors (topologies, training strategies, and data distributions) on convergence, we utilize five widely used machine learning models as local models. Specifically, we select two convex models (SVM and Logistic Regression) and three non-convex models (ResNet, DistilBERT, and MiniGPT-4). Our experimental results demonstrated that all models, across different topologies and training strategies, achieved convergence under IID

data. Additionally, we investigated the effect of varying non-IID levels on model convergence.

1.2 Thesis Outline

This thesis is divided into five parts. Besides the first part, which is the introduction 1, the second part presents the background 2 knowledge relevant to this work and important research related to our study. In the third part, we perform a detailed mathematical analysis 3 of the convergence of each DFL model. The fourth part provides a detailed analysis of the convergence of different DFL models under Non-IID data through two sets of experiments 4. In the final part, we discuss future research directions and conclude the thesis 5.

Chapter 2

Background and Related Work

In this chapter, we begin by providing the essential context needed to understand the content presented in this thesis. Subsequently, we conduct a thorough examination of existing literature relevant to our research.

2.1 Background

2.1.1 Centralized Federated Learning

Centralized Federated Learning (CFL) is a machine learning paradigm first proposed in 2016 [27]. As shown in Figure 2.1, there is a central server and multiple user clients. The first step in the training process is for the server to send the initial model parameters to each client. Subsequently, each client conducts independent training on local data based on these initial parameters. After the training is completed, clients upload the updated model parameters to the central server, which is responsible for aggregating these parameters, generating the global model and distributing it back to the client.

Advantage of centralized federal learning is that it can effectively protect user privacy, because during the whole process, data from clients is always stored locally and does not need to be shared with the server. This makes CFL highly practical in privacy-sensitive applications. However, with the increase in the number of clients, the communication cost increases accordingly, which may affect the overall efficiency of the system. In addition, the architecture of CFL relies on the normal operation of the central server. If the central server fails, the whole training process will not continue. The single-point failure risk of this centralized architecture is an important defect of CFL, so in practical applications, additional fault tolerance mechanisms are often needed to ensure the reliability of the system [46].

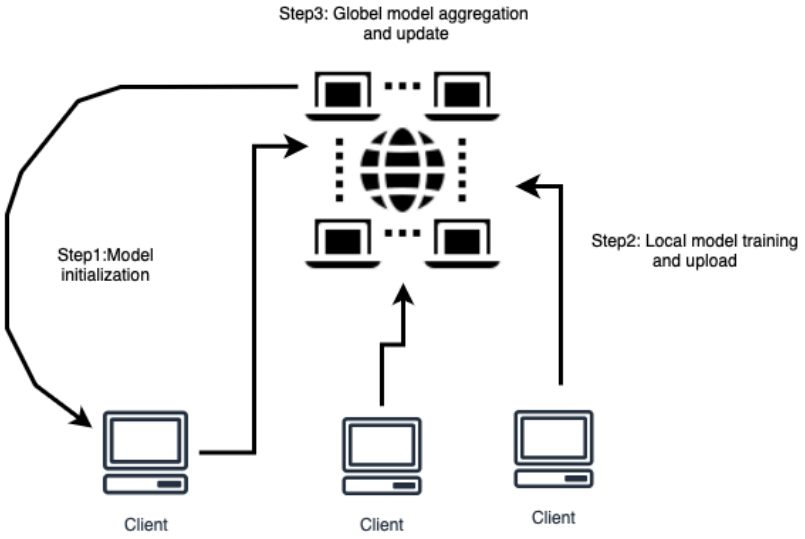

Figure 2.1: Centralized federated learning architecture.

2.1.2 Decentralized federated learning

Decentralized Federated Learning (DFL) is a distributed machine learning paradigm in which there is no single central server. Instead, model updates and parameter aggregation are carried out through direct communication between multiple devices. Each device independently trains a model on its local data and exchanges model updates with other devices through a specific network topology [45].

DFL can be applied across a broad spectrum of pertinent scenarios, including emerging fields such as healthcare, smart cities, Industry 4.0, and mobile services. It proves beneficial in addressing challenges like meta-verse, enhancing urban planning, and optimizing service provision [3]. DFL excels in enabling the secure sharing of privacy-preserving data between real and virtual entities. This capability proves particularly advantageous in fostering task coordination and asynchronous knowledge exchange within domains such as robotics, energy, and utilities sectors, with significant applications in Industry 4.0 and mobile services.

In the real world, two primary types of DFL exist: *cross-silo DFL* and *cross-device DFL* [45] [3]. Cross-silo DFLs are established by organizations or data centers, typically featuring a relatively small number of nodes. On the other hand, cross-device DFLs involve a larger number of nodes, each with a modest amount of data and limited computational power. Within each set of nodes, specific roles are assigned,

including trainer, aggregator, proxy, and idle [3]. The trainer is the node used for training.

The aggregator is the node used for aggregating parameters sent from different devices. Proxy nodes act as intermediaries in DFL, typically responsible for forwarding or aggregating data between different devices. Idle nodes are those that temporarily do not participate in the current training task. These nodes may refrain from executing training or aggregation tasks during a certain period due to resource limitations, task scheduling, or other reasons.

In our study, we explored various network topologies and aggregation methods, such as ring topology, linear topology, full mesh topology, and star topology. In the continuous ring and linear topologies, each node acts as a trainer. In the aggregate ring, aggregate linear, and mesh topologies, each node serves as both a trainer and an aggregator. In the star topology, the central node functions as both the trainer and aggregator, while the other nodes only serve as trainers.

It is noteworthy that we did not designate proxy or idle nodes in our study because we aimed to investigate the impact of different topologies and training strategies on the performance of DFL.

In section 3.1, we show our mathematical analysis is applicable to both cross-silo and cross-device DFL because we define N as the total number of clients, and N can be any number. This implies that our analysis can be applied to scenarios with varying numbers of devices. In the section 4, we show our experiments also apply to both scenarios because we employed various local models, simulating the needs of different situations in real life.

2.1.3 Non-IID Data

A significant challenge in DFL lies in the Non-IID data distribution, a factor widely recognized by various researchers. Non-IID data refers to the uneven or random distribution of data on different clients in statistical nature. In federal learning, this means that the data on each device may vary significantly in statistical attributes [47] [49]. The training data on each client in DFL is heavily influenced by the specific local devices they use [45] [3]. As a result, the data distribution among connected clients can vary significantly. Zhao et al. [47] have demonstrated that the accuracy of

FedAvg [27], a fundamental algorithm in federated learning that averages the model parameters from each client weighted by the number of samples on the device to form a global model, diminishes notably with increasing data heterogeneity.

Hangyu et al. [49] have showcased various types of non-IID data in real-life scenarios, categorized into "attribute skew" and "label skew." "Attribute skew," often associated with Horizontal Federated Learning, involves variations in the number of attributes on different devices. Notably, our study predominantly explores "label skew" due to its prevalence in vertical federated learning, which is more relevant to our study on DFL. "Label skew" refers to differences in the number of labels for each device. In our investigation, we quantify the degree of non-Independent and Identically Distributed (NON-IID) Data in Section 4.

2.1.4 Network Topology

Since DFL uses P2P transmission, common network topologies in real-life scenarios can be applied to DFL [5]. The works [45] [3] [9] show that network topology can affect the efficiency of DFL. In detail, ring and linear structures are similar [34] [42] [44]. Specifically, when one device finishes its operation, it passes the parameters to the next device. This structure allows for sequential parameter transmission, enabling each device to update the model in turn, resulting in low communication overhead and high computational efficiency, making it suitable for environments with limited bandwidth.

In the case of the star model, the central node is responsible for parameter transmission and aggregation among all devices. We first select one device as the central node, and then all other devices send their parameters to the central node. The central node aggregates these parameters and sends the updated parameters back to each device [45], [9].

We want to point out that in CFL, the central is a server, while in the star topology of DFL, the central node can function as both a server and a client. Users can perform both training and aggregation operations on the central node, meaning that the central node can handle data as well as perform computations. This flexibility allows users to choose the most suitable device as the central node, such as the one with the strongest communication capabilities and the highest computational power.

For the mesh model, all devices are interconnected, allowing for direct parameter transmission and aggregation [25], [33], [5]. This structure offers high redundancy and robustness, as it can tolerate failures of some devices without affecting the entire system's operation. However, the communication complexity is high, and management becomes more complicated.

In summary, different network topologies can be applied to DFL based on actual needs and resource conditions to optimize system performance and efficiency. In our study, we analyze four types of network topologies: linear, ring, star, and mesh, under NON-IID data conditions. This is because we believe that since DFL relies on network transmission, traditional network structures should be applicable to various types of DFL.

2.1.5 Training Strategy

There are two main Training strategy for DFLs: aggregation and continuous.

Aggregation for DFL is proposed by Yuan et al. [45]. They assume that there is a method similar to FedAvg [27] that can aggregate the parameters of different devices. The device does not rely on a single parameter but aggregates multiple parameters according to a certain weight to obtain new parameters, and trains based on these new parameters. We use a method similar to FedAvg for aggregation, where the proportion of the number of samples in the total number of samples in a device data set is used as the aggregation parameter. For example, suppose there are two devices participating in the training process. Device A has 200 samples, Device B has 700 samples. The total number of samples is 1000. The aggregation weights for the parameters from these devices would be 0.2 for Device A, 0.7 for Device B. Again, suppose Device A has 300 samples, Device B has 700 samples, and the total number of samples is 1000. The aggregation weights for the parameters from these devices would be 0.3 for Device A, 0.7 for Device B. Therefore, Device B's parameters will have a greater influence on the final aggregated parameters due to its larger proportion of the total data.

Continuous training, in this mode, devices sequentially receive and update model parameters. Each device trains using its local data and then passes the updated parameters to the next device. The advantage of this mode is its simplicity and

directness. For example, like the figure 3.1, client1 sends its parameters to client2. Client2 then uses these parameters as the initial parameters for its own training. After training, client2 sends its updated parameters to client3, and so on.

It focuses on the ability of a model to continuously learn and adapt to new data without forgetting previously learned information [45] [2]. This is particularly important in DFL scenarios where data distribution can change over time or where models need to be frequently updated with new data from network nodes.

Both aggregation and continual learning are critical for the effective implementation and optimization of DFL systems, ensuring they remain robust and efficient across varying network topologies and evolving data distributions. We combine the network topology 2.1.4 and training strategy, we get the 6 DFL topology. We show the six combinations used in this study on table 2.1. It is important to note that continuous star and continuous mesh structures are impractical in real-life scenarios. If we use the continuous approach in the star topology, the central client receives multiple parameters from different devices and does not know which parameter to use as the initial parameter for training. Similarly, in the mesh structure, each device receives parameters from all other devices and faces the same issue of not knowing which parameter to base its training on,in the other word,it does know which parameter to use as the initial parameter for training. We will show the formulation of different types of aggregation methods in Section 3.1.

DFL	**Training Mode**
Continuous Ring	Continuous
Continuous Linear	Continuous
Full Mesh	Aggregate
Star	Aggregate
Aggregate Ring	Aggregate
Aggregate Linear	Aggregate

Table 2.1: The six combinations of DFL topologies and training stratey used in this study.

2.1.6 Strongly Convex and L-smooth

Several papers have utilized theoretical analysis in Federated Learning (FL) by assuming that the local Machine Learning (ML) models on devices are strongly convex

and L-smooth [23] [22] [29]. This assumption facilitates the derivation of convergence rates for FL algorithms. Strong convexity and L-smoothness are mathematical conditions that describe the curvature of the functions being optimized:

Strongly Convex: A function is strongly convex if there is a constant $\mu > 0$ such that for all points x and y in its domain, the function lies above a quadratic curve with curvature μ. This property ensures that the function has a unique global minimum and that it behaves well for optimization algorithms, leading to faster convergence rates [41].

L-smooth: A function is called L-smooth if its gradient is L-Lipschitz continuous. This means that the rate of change of the function is constrained by L; in other words, the change in the gradient is bounded by L times the distance between x and y. The benefit of this is that it ensures that optimization algorithms do not overshoot the minimum during their steps. This is a common assumption in optimization. These assumptions are crucial in the analysis of federated learning algorithms because they allow researchers to apply optimization theory to analyze the behavior of federated learning algorithms and to predict the speed and reliability with which these algorithms converge to a good solution. [4]

2.2 Related Works

2.2.1 Centralized Federated Learning

Hangyu et al. [49] demonstrated various types of non-IID data in real-world scenarios and proved through experiments that such data significantly reduces the prediction accuracy of the global model. Their research emphasized that non-IID data in centralized federated learning leads to a substantial decline in model performance. Zhao et al. [47] further proved that as data heterogeneity increases, the accuracy of the FedAvg algorithm decreases significantly. Their findings show that when data distribution differs greatly among different clients, the effectiveness of centralized federated learning is greatly compromised.

Li et al. [23] mathematically proved that the higher the degree of non-IID, the worse the convergence of centralized federated learning. Their theoretical analysis provides a solid foundation for understanding the impact of non-IID data on the convergence of federated learning. However, their work lacks analysis of different network topologies and only focuses on the FedAvg training strategy, without exploring various other training strategies.

While these studies reveal the significant impact of non-IID data on centralized federated learning, most existing research primarily focuses on CFL and seldom discusses the performance of DFL under non-IID data, different training strategies, and various network topologies. Our research aims to fill these gaps by experimentally and mathematically analyzing the performance of DFL composed of different network structures under varying degrees of non-IID.

2.2.2 Decentralized Federated Learning

In the field of decentralized federated learning (DFL), many studies have explored the performance of various DFL methods. Sheller et al. [34] were the first to analyze the performance comparison between continuous linear topology, continuous ring topology, and centralized federated learning. They conducted experiments with the same number of epochs and provided a detailed comparison of these methods. This study offers valuable insights into the impact of different topologies on DFL performance. However, their work lacks analysis of star and mesh models, and they did not explore

the performance of DFL under non-IID data, nor did they provide theoretical proofs.

Another study [22] designed a DFL method that combines CFL with continuous linear topology and found that this new method performs better under non-IID data. This indicates that combining centralized and decentralized methods can achieve better performance when handling non-IID data. However, they did not analyze DFL under different network topologies.

Angelia et al. [29] analyzed the network topologies of DFL using undirected and directed graphs. They applied the Metropolis algorithm to update directed graphs and the Push-sum algorithm to update undirected graphs, analyzing the computational complexity and convergence speed of each method. However, their study did not consider the impact of non-IID data, nor did they provide detailed analysis of specific commonly used network topologies. Additionally, their research focused more on mathematical analysis, lacking experimental validation.

In summary, although existing studies have made some progress in exploring DFL, there are still many unresolved issues. Particularly in handling non-IID data and complex network topologies, the performance and convergence of DFL require further research and optimization. In our work, we explore the performance of DFL through both theoretical 3.1 and experimental 4 approaches. We first define six different DFL configurations based on two different training strategies and network topologies and conduct convergence analysis on them 3.1. Then we define the required non-IID settings in the experiments 4and measure the performance of the six different DFL configurations under varying degrees of non-IID.

Chapter 3

Performance Analysis of DFL Deployments

3.1 Convergence Rate Analysis

This section will develop the objective functions for the six different DFL deployments using the notations presented in Table 3.1.

Table 3.1: Notation Table

Symbol	Description
$\mathcal{L}oss(\cdot;\cdot)$	User-specified loss function
p_k	Weights of the k-th device
N	Number of samples
$F(w)$	Function for Decentralized Federated Learning (DFL)
F^*	The optimal F under ideal conditions
F_k	Objective for the k-th device
F_k^*	The optimal F_k under ideal conditions
ξ^k	SGD randomly chosen sample from k-th device
w^k	Weights for k-th device
n_k	Number of samples associated with the k-th device
L	Parameter for L-smoothness
μ	Parameter for μ-strong convexity
Z	The level for the NON-IID

3.1.1 Network Environment

To simplify the problem and focus on the impact of network topology on the convergence of decentralized federated learning (DFL), we assumed that each device has the same communication and computing capabilities. Additionally, we assume a reliable communication channel with no packet loss. Furthermore, we do not distinguish between wired and wireless communication, instead assuming an abstract mode of communication. This abstraction enables us to concentrate on the core effects of network topology without being constrained by specific communication technologies.

3.1.2 Problem Formulation

We formulate the distributed optimization model for Decentralized Federated Learning (DFL) deployment as in Equation 3.1. For the function 3.1, regardless of the type of decentralized federated learning (DFL) model, their goal is consistent: to optimize the global model parameters by minimizing the global loss function $F(w)$. Different network topologies (such as linear, ring, star, etc.) determine the way information is transmitted between devices, but all models follow the same core objective. This objective is to collaboratively compute local loss functions and share information to jointly optimize the global model.

$$\min_{w} \left\{ F(w) = \sum_{k=1}^{N} p_k F_k(w) \right\} \tag{3.1}$$

Where p_k denotes the weights of the k-th device ($p_k \geq 0$, $\sum_{k=1}^{N} p_k = 1$), N is the total number of devices, and w signifies the model parameters.

The objective function for each device k (where $1 \leq k \leq N$) using the entire dataset $\mathbf{x}_k = \{x_{k,1}, \ldots, x_{k,n_k}\}$ is expressed as:

$$F_k(w) = \frac{1}{n_k} \sum_{j=1}^{n_k} \mathcal{L}oss(w; x_{k,j}) \tag{3.2}$$

where $\mathcal{L}oss(\cdot;\cdot)$ represents a user-specified loss function and n_k represents the number of samples associated with the k-th device.

If we uniformly select ξ^k samples from device k's dataset $\mathbf{x}_k$, the objective function will then be:

$$F_k(w) = \frac{1}{n_k} \mathcal{L}oss(w; \xi^k) \tag{3.3}$$

3.1.3 Assumptions

We use the same assumptions as Li et al. [23]. $F_1, \ldots, F_N$ are all L-smooth and μ-strongly convex, ensuring that the gradient does not change too rapidly and the value of the function between any two points is not only above the tangent line but also above the tangent line plus a positive term proportional to $\|w - w'\|^2$. For $k = 1, \ldots, N$, we also assume:

$$E\|\nabla F_k(w^k, \xi^k) - \nabla F_k(w^k)\|^2 \leq \sigma_k^2$$

and

$$E\|\nabla F_k(w^k, \xi^k)\|^2 \leq G^2.$$

These assumptions indicate that the noise's impact on gradient estimation is limited. We consider F_k^* to be the local optimal solution for the DFL model, in the other word, it is the minimum values of F_k. We use the term

$$F^* = \frac{1}{N}\sum_{k=1}^{N} F_k^*$$

and

$$Z = F^* - F_k^*$$

to quantify the degree of non-IID. If the data is IID, in the theoretical context, $Z = 0$.

3.1.4 Converge Analysis

We show that six different DFL converges to the optimal global solution for strongly convex functions on Non-IID data. This means we are interested in evaluating $E\left(F\left(\bar{w}\right)\right) - F\left(w^*\right)$. Given that F_k is L-smooth, we can derive the following inequality:

$$F(\bar{w}) \leq F(w^*) + \langle \nabla F(w^*), \bar{w} - w^* \rangle + \frac{L}{2}\|\bar{w} - w^*\|^2 \tag{3.4}$$

Due to the monotonicity of the expected value, we have:

$$E(F(\bar{w})) \leq E(F(w^*)) + E\left(\langle \nabla F(w^*), \bar{w} - w^* \rangle\right) + \frac{L}{2}E\|\bar{w} - w^*\|^2$$

Subtracting F_k^* from both sides yields:

$$\begin{aligned} E(F(\bar{w})) - F^* \leq E(F(w^*)) - F^* + \\ E\left(\langle \nabla F(w^*), \bar{w} - w^* \rangle\right) + \frac{L}{2}E\|\bar{w} - w^*\|^2 \end{aligned} \tag{3.5}$$

We have the function: $E\left(F\left(w^*\right)\right) - F^* = 0$.

Because strongly convex functions have the property that their gradient is zero at the optimal point w^*, when $F(w)$ reaches its optimal solution w^*, the gradient $\nabla F(w^*) = 0$. In detail, this is because the gradient vector points to the direction of the fastest growth of the function, so at the most advantage (whether local optimal or global optimal), the growth rate of the function should be zero, which means that the gradient is also zero.

Figure 3.1: Continuous linear.

Thus, the $(\langle \nabla F(w^*), w - w^* \rangle) = 0$. We get the optimal function for DFL:

$$E(F(\bar{w})) - F(w^*) \leq \frac{L}{2} E||\bar{w} - w^*||^2 \tag{3.6}$$

This function (3.6) inequality indicates that in DFL, the expected difference in loss values between the obtained model parameters $\bar{w}$ and the optimal model parameters w^*, denoted as $E(F(\bar{w})) - F(w^*)$, can be bounded by the expected squared distance between them, $\frac{L}{2} E\|\bar{w} - w^*\|^2$. In other words, this inequality shows that the smaller the distance between the average model parameters and the optimal parameters, the smaller the error in the loss function values.

Our subsequent steps will involve simplifying $\|\bar{w} - w^*\|^2$ based on different DFL structures. By doing so, we will derive different convergence formulas for each DFL topology.

3.1.4.1 Continuous Linear

Continuous linear refers the network topology 2.1.4 as linear and the training strategy 2.1.5 as continuous.

The Equation(3.7) shows how the continuous linear transfer the parameter. In detail, the continuous linear DFL [34] is structured as a line, with each client serving as a node connected to the next. For each device k, it receives the model parameters from the previous device $k-1$ and updates the model using its local dataset ξ^k. Subsequently, it forwards the trained parameters along the line to the next client. This sequential and iterative training continues along the line, fostering collaboration among the clients to collectively enhance the learning and performance of the entire system. This process can be formulated as the following equation:

$$w^k = w^{k-1} - \eta_k \nabla F_k \left(w^{k-1}, \xi^k\right) (1 \leq k \leq N) \tag{3.7}$$

where w^k denotes the updated parameter for device k, η_k is the learning rate of device k and $\nabla F_k\left(w^{k-1}, \xi^k\right)$ represents the gradient of the local objective function $F_k\left(w^{k-1}, \xi^k\right)$ for device k based on the previous device's weight w^{k-1} and its local data set ξ^k.

We set the w^k as the parameter which trained on k device. Thus, since local model on each device is totally same, it can be seem as a big model, we suppose the $A_k = \nabla F_k\left(w^{k-1}, \xi^k\right), \bar{A}_k = \nabla F_k\left(w^{k-1}\right)$; therefore, $EA_k = \bar{A}_k$, $w^k = w^{k-1} - \eta_k A_k$. Thus, the function (3.6) is equal to:

$$\begin{aligned} ||w^k - w^*||^2 &= ||w^{k-1} - \eta_k A_k - w^* - \eta_k \bar{A}_k + \eta_k \bar{A}_k||^2 \\ &= ||w^{k-1} - \eta_k \bar{A}_k - w^*||^2 + 2\eta_k < w^{k-1} - w^* - \eta_k \bar{A}_k, \\ &\qquad \bar{A}_k - A_k > +\eta_k^2 ||A_k - \bar{A}_k||^2 \end{aligned} \tag{3.8}$$

because the $EA_k = \bar{A}_k$, thus, the expectation of $2\eta_k < w^{k-1} - w^* - \eta_k \bar{A}_k, \bar{A}_k - A_k >= 0$.

So, we will bond for the $||w^{k-1} - \eta_k \bar{A}_k - w^*||^2$ first.

$$\begin{aligned} ||w^{k-1} - \eta_k \bar{A}_k - w^*||^2 &= ||w^{k-1} - w^*||^2 \\ -2\eta_k < w^{k-1} - w^*, \bar{A}_k > &+ \eta_k^2 ||\bar{A}_k||^2 \end{aligned} \tag{3.9}$$

since $\eta_k^2 ||\bar{A}_k||^2 = \eta_k^2 ||\nabla F_k\left(w^k\right)||^2$, and because the L-smooth,

$$\eta_k^2 ||\nabla F_k\left(w^k\right)||^2 \leq 2\eta^2 L\left(F_k\left(w^k\right) - F_k^*\right). \tag{3.10}$$

Then, we bonding at the $-2\eta_k < w^{k-1} - w^*, \bar{A}_k >$, because the local model is strongly convex, which mean,

$$\begin{aligned} -2\eta_k < w^{k-1} - w^*, \nabla F_k\left(w^{k-1}\right) > \\ \leq -2\eta_k (F_k\left(w^{k-1}\right) - F_k\left(w^*\right) - \frac{\mu}{2} \parallel w^{k-1} - w^* \parallel_2^2) \end{aligned} \tag{3.11}$$

Then, we substitute above two equation into Equation 3.9:

$$\begin{aligned} ||w^{k-1} - \eta_k \bar{A}_k - w^*||^2 \leq ||w^{k-1} - w^*||^2 - 2\eta_k (F_k\left(w^{k-1}\right) \\ -F_k\left(w^*\right) - \frac{\mu}{2} \parallel w^{k-1} - w^* \parallel_2^2) + 2\eta_k^2 L\left(F_k\left(w^k\right) - F_k^*\right) \end{aligned} \tag{3.12}$$

The Equation 3.9 is equal to:

$$\begin{aligned} ||w^{k-1} - \eta_k \bar{A}_k - w^*||^2 \leq ||w^{k-1} - w^*||^2 - 2\eta_k [F_k\left(w^{k-1}\right) \\ -F_k\left(w^*\right) - \frac{\mu}{2} \parallel w^{k-1} - w^* \parallel^2] + (2\eta_k)^2 L\left(F_k\left(w^k\right) - F_k^*\right) \\ \leq ||w^{k-1} - w^*||^2 - 2\eta_k F_k\left(w^{k-1}\right) + 2\eta_k F_k\left(w^*\right) \\ +\mu\eta_k \parallel w^{k-1} - w^* \parallel^2 + (2\eta_k)^2 L\left(F_k\left(w^k\right) - F_k^*\right) \\ = (1 + \mu\eta_k) ||w^{k-1} - w^*||^2 - 2\eta_k \left(F_k\left(w^{k-1}\right) - F_k^*\right) \\ +(2\eta_k)^2 L\left(F_k\left(w^k\right) - F_k^*\right) \end{aligned} \tag{3.13}$$

Then, we let the $J = -2\eta_k \left(F_k\left(w^{k-1}\right) - F_k^*\right) + (2\eta_k)^2 L\left(F_k\left(w^k\right) - F_k^*\right))$.

Since $F_k^* - F_{k-1} \leq F^* - F_{k-1} \leq F^* - F_k^*$; therefore, $J \leq 2\eta_t \left(F^* - F_k^*\right) + 2\eta_t^2 L \left(F_k - F_k^*\right)$. To bond the $(F_k - F_k^*)$. Because the F_k is L-smooth, we can get $(F_k - F_k^*) \leq \frac{L}{2} \parallel W^k - W^* \parallel^2 + \left(W^k - W^*\right)^T \nabla F_k\left(W^*\right)$. Since $\nabla F_k\left(W^*\right) = 0$, $J \leq 2\eta_k Z + \eta_k^2 L^2 \parallel w^k - w^* \parallel^2$, equation 3.13 is become to $(1 + \mu\eta_k) ||w^{k-1} - w^*||^2 + 2\eta_k Z + \eta_k^2 L^2 \parallel w^k - w^* \parallel^2$. Therefore, the equation 3.8 becomes:

$$\begin{aligned} \parallel w^k - w^* \parallel^2 \leq (1 + \mu\eta_k) ||w^{k-1} - w^*||^2 \\ +2\eta_k Z + \eta_k^2 L^2 \parallel w^k - w^* \parallel^2 + \eta_t^2 \parallel A_t - \bar{A}_t \parallel^2 \end{aligned} \tag{3.14}$$

And taking expectation on both sides of Equation 3.14

$$\begin{aligned} E \parallel w^k - w^* \parallel^2 \leq (1 + \mu\eta_k) E||w^{k-1} - w^*||^2 \\ +2\eta_k Z + \eta_k^2 L^2 E \parallel w^k - w^* \parallel^2 + \eta_k^2 E \parallel A_k - \bar{A}_k \parallel^2 \end{aligned} \tag{3.15}$$

We next step is bond the $\eta_t^2 E \parallel A_t - \bar{A}_t \parallel^2$, by the assumption, $\eta_t^2 E \parallel A_t - \bar{A}_t \parallel^2 \leq \eta_t^2 \sigma_k^2$.

Therefore, we can see that in the final Function 3.16, if the Z (degree of non-IID) approaches zero, the left part will be bounded by a constant. This means that when the data distribution on each client is the same, the DFL model will converge. However, as the level of non-IID increases, the value on the right side of the equation increases, causing the gap between the actual loss and the ideal loss to increase. This leads to the model becoming more divergent.

$$E\left[F\left(w^k\right) - F^*\right] \leq \frac{L}{2} \left[\left(1 + \mu\eta_k + \eta_k^2 L^2\right) E||w^0 - w^*||^2 + 2\eta_k Z + \eta_k^2 \sigma_k^2\right] \tag{3.16}$$

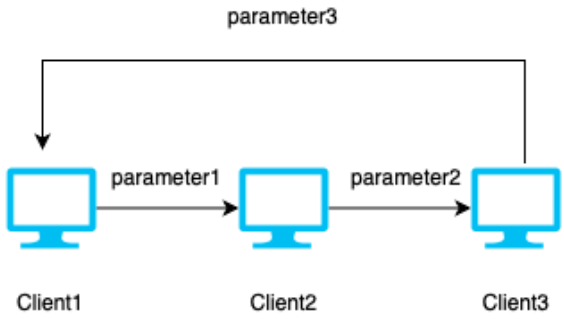

Figure 3.2: Continuous ring.

3.1.4.2 Continuous Ring

The continuous ring DFL [34] comprises a network of devices interconnected in the form of a ring. We want to mention that the continuous ring is very similar with continuous linear because there have the same training strategy 2.1.5.

Defined d as the cumulative number of times the model parameters are passed in all devices, which is an indicator of the flow of the model throughout the network, where $d \in 1, ..., N * t$. Assuming that device1 received initial parameters w_0, in each training round t ($t > 0$), the k-th ($k = d - (t-1)\,N$) device updates the initial model parameters using its data and then transmits the updated parameters to the $(k+1)$-th device.

The subsequent device considers the received model parameters from the previous device as the initial model and uses its data to update it. When the model is transmitted to the last device, that device sends the trained parameters back to the first device. Subsequently, the first device sets these parameters as the initial values and initiates training again. This cyclic process is mathematically represented by the following equation:

$$\begin{aligned} w^d = w^{d-1} - \eta_d \nabla F_d \left(w^{d-1}, \xi^d\right) \\ k = d - (t-1)\,N \end{aligned} \tag{3.17}$$

Here, t represents the number of complete cycles from the first client to the last client and then back to the first client, constituting one full training cycle. Each complete cycle allows every device an opportunity to update and pass along the

Figure 3.3: Aggregate Linear.

model parameters, considered as one complete training round. Therefore, the coverage analysis of ring continuous is similar with the linear continuous.

$$E\left[F\left(w^{d}\right)-F^{*}\right] \leq \frac{L}{2}\left[\left(1+\mu\eta_{d}+\eta_{d}^{2}L^{2}\right)E||w^{0}-w^{*}||^{2}+2\eta_{d}Z+\eta_{d}^{2}\sigma_{d}^{2}\right] \tag{3.18}$$

Since C_ring and C_linear use the same aggregation strategy, their mathematical convergence analysis formulas are similar. Therefore, the derivation process is omitted. By observing the formula, we can see that as Z (the degree of non-IID) increases, the value on the left side becomes larger, and the gap between the optimal solution and the actual value on the right side also increases, causing the model to become more divergent.

3.1.4.3 Aggregate Linear

Aggregate linear DFL refers to the use of a linear network topology 2.1.4 combined with an aggregation training strategy 2.1.5. Similar to the continuous linear, this deployment comprises a series of interconnected devices aligned linearly. As the Figure 3.3, each device within this network possesses a unique dataset. After training on Device 1, it transmits its parameters to Device 2.

Following the same procedure as the continuous linear approach, Device 2 trains based on the parameters from Device 1. Device 2 then transmits both its own parameters and those from Device 1 to Device 3. Device 3 aggregates the parameters from both Device 1 and Device 2, and then trains based on the aggregated parameters. As previously mentioned in Section 2.1.5, the aggregation follows a method similar to FedAvg.

Specifically, we calculate the cumulative sample count of Devices 1 and 2, termed

as s_{sum}. Additionally, we denote the individual sample counts for Device1 and Device2 as $s1$ and $s2$, respectively. The final parameter set transmitted from Device2 to Device3 is a weighted combination ($w1$ is the weight from client1 to client2, $w2$ is the weight after client2 trained its own dataset): $\frac{s1}{s_{sum}} \times W_1 + \frac{s2}{s_{sum}} \times W_2$.

This process is mathematically represented by the following equation; when k is at most 2:

$$w^k = w^{k-1} - \eta_k \nabla F_k \left(w^{k-1}, \xi^k \right) \tag{3.19}$$

When k is larger than 2:

$$\begin{aligned} w^k = & \left(\frac{\left(s_{k-1}\right) w^{k-1} + \left(s_{k-2}\right) w^{k-2}}{\sum_{i=1}^{k-1} s_i} \right) - \eta_k \nabla F_k \\ & \left(\left(\frac{\left(s_{k-1}\right) w^{k-1} + \left(s_{k-2}\right) w^{k-2}}{\sum_{i=1}^{k-1} s_i} \right), \xi^k \right) \end{aligned} \tag{3.20}$$

Next, we will analyze its convergence: For the $w^k = M^k - \eta_k \nabla F_k \left(M^k, \xi^k \right)$ and $M^k = \left(\frac{s_{k-1} w^{k-1} + s_{k-2} w^{k-2}}{\sum_{i=k}^{k-1} s_i} \right)$ we let $H = \nabla F_k \left(M^k, \xi^k \right)$ and $\bar{H} = \nabla F_k \left(M^k \right)$ Thus, $EH = \bar{H}$ and $w^k = M - \eta_k H$.

$$\begin{aligned} \parallel w^k - w^* \parallel^2 = & \parallel M - \eta_k H - w^* - \eta_k \bar{H} + \eta_k \bar{H} \parallel^2 \\ = & \parallel M - \eta_k \bar{H} - w^* + \eta_k \bar{H} - \eta_k H \parallel^2 \end{aligned} \tag{3.21}$$

The same with the continuous linear, Equation 3.6 is equal to $|M - \eta_k \bar{H} - w^*\|^2 - 2 < M - \eta_k \bar{H} - w^*, \eta_k \bar{H} - \eta_k H > + \eta_k^2 \parallel \bar{H} \parallel^2$ and we will bond the $|M - \eta_k \bar{H} - w^*\|^2$ first. $\|M - \eta_k \bar{H} - w^*\|^2 = \|M - w^*\|^2 - 2\eta_k < M - w^*, \bar{H} > + \eta_t^2 \|\bar{H}\|^2$
and we will bond the $2\eta_k^2 < M - w^*, \bar{H} >$ and $\eta_k^2 \|\bar{H}\|^2$

$$\begin{aligned} 2\eta_k^2 < M - w^*, \bar{H} > \leq & -2\eta_k \left(F_k \left(M \right) \right. \\ & -F \left(w^* \right) + \frac{\mu}{2} \|M - w^*\|_2^2 \end{aligned} \tag{3.22}$$

$$\eta_k^2 \|\nabla F_k \left(M \right) \|^2 \leq 2\eta_k^2 L \left(F_k \left(M \right) - F_k^* \right) \tag{3.23}$$

Then, we get

$$\begin{aligned} \|M - \eta_k \bar{H} - w^*\|^2 \leq & \left(1 + \mu\eta_k\right) \|M - w^*\|^2 + 2\eta_k \left(F_k \left(w^* \right) \right. \\ & -F_k \left(M \right) + 2\eta_k^2 L \left(F_k \left(M \right) - F_k^* \right) \end{aligned} \tag{3.24}$$

The same way as the continue linear, we can got the equation:

$$\begin{aligned} A \leq (1+\mu\eta_k)\left\|M-w^*\right\|^2 + 2\eta_k Z + \eta_k^2 L^2 \|M-w^*\|^2 \\ \left\|w^k - w^*\right\|^2 \leq (1+\mu\eta_k)\left\|M-w^*\right\|^2 \\ +2\eta_k Z + \eta_k^2 L^2 \|M-w^*\|^2 + \eta_k^2\sigma_k^2 \end{aligned} \tag{3.25}$$

By the strong convex, we can instead the $M^k = \left(\frac{s_{k-1}w^{k-1}+s_{k-2}w^{k-2}}{\sum_{i=k}^{k-1} s_i}\right)$ to $V^k = \left(\frac{s_{k-1}w^0+s_{k-2}w^0}{\sum_{i=k}^{k-1} s_i}\right)$

Thus, we get the function for aggregate linear:

$$\begin{aligned} E(F(\bar{w})) - F\left(w^*\right) \leq \frac{L}{2}[(1+\mu\eta_k)\left\|V^k - w^*\right\|^2 \\ +2\eta_k Z + \eta_k^2 L^2 \|V^k - w^*\|^2 + \eta_k^2\sigma_k^2] \end{aligned} \tag{3.26}$$

We see that all parameters except Z are constants, which means that the right side of the formula is only affected by Z. However, an increase in Z will cause the right side to increase as a whole. This simultaneously affects the value of the left side of the formula, leading to poorer overall convergence of the model.

3.1.4.4 Aggregate Ring

The aggregate ring is similar to the aggregate linear because their have same training strategy. A series of interconnected devices is comprised of a ring. Each device possesses a unique dataset. There is only one difference. As the Figure 3.4, when running to the last device, the last aggregated parameters will be sent back to the first device.

This process is mathematically represented by the following equation; when k is at most 2.

$$w^d = w^{d-1} - \eta_d \nabla F_d\left(w^{d-1}, \xi^d\right) \tag{3.27}$$

when k is larger than 2

$$\begin{aligned} w^d = \left(\frac{\left(s_{d-1}\right)w^{d-1} + \left(s_{d-2}\right)w^{d-2}}{\sum_{i=1}^{d-1} s_i}\right) - \eta_d \\ \nabla F_d\left(\left(\frac{\left(s_{d-1}\right)w^{d-1} + \left(s_{d-2}\right)w^{d-2}}{\sum_{i=1}^{d-1} s_i}\right), \xi^d\right) \\ k = d - (t-1)\,N \end{aligned} \tag{3.28}$$

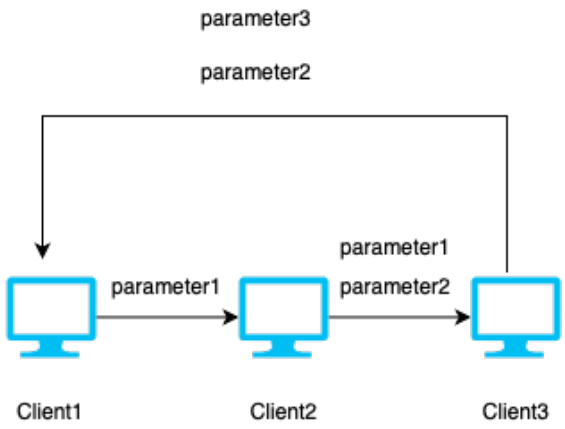

Figure 3.4: Aggregate Linear

Because the aggregate ring have same training strategy:

$$E(F(\bar{w}))-F\left(w^{*}\right) \leq \frac{L}{2}[(1+\mu\eta_d)\left\|V^{d}-w^{*}\right\|^{2}+2\eta_d Z+\eta_d^{2}L^{2}\left\|V^{d}-w^{*}\right\|^{2}+\eta_d^{2}\sigma_d^{2}] \quad (3.29)$$

Since A_ring and A_linear use the same aggregation strategy, their mathematical convergence analysis formulas are similar. Therefore, the derivation process is omitted. By observing the formula, we can see that as Z (the degree of non-IID) increases, the value on the left side becomes larger, and the gap between the optimal solution and the actual value on the right side also increases, causing the model to become more divergent.

3.1.4.5 Aggregate Star

The aggregate star is similar to CFL [23, 27], however, like the Figure 3.5, the center device no longer only plays the role of a server, instead, it can participate in the training of local models. First, select a central device as the center of the whole deployment. After that, the initial parameters are issued by the central device to each device, and then each device (including the central device) is trained based on the initial parameters. After that, the central receives the parameters of all devices for aggregation. This process is then iteratively repeated.

This process is mathematically represented by the following equation:

$$w^{k}=\sum_{i=1}^{n}\left(\frac{s_i w_i}{\sum_{j=1}^{n} s_j}\right)-\eta_k \Delta F_k\left(\sum_{i=1}^{n}\left(\frac{s_i w_i}{\sum_{j=1}^{n} s_j}\right), \xi^{k}\right) \quad (3.30)$$

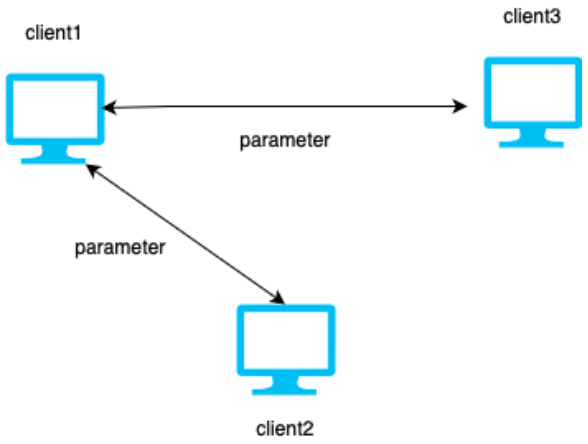

Figure 3.5: Star

To analyze the coverage, the aggregate star and mesh DFL is similar with CFL, Li et al. [23] have proved it as: $\mathbb{E}[F(w_T)] - F^* \leq \frac{2\kappa}{\gamma+T}\left(\frac{B}{\mu} + 2L\|w_0 - w^*\|^2\right), B = \sum_{k=1}^{N} p_k^2\sigma_k^2 + 6LZ + 8(E-1)^2G^2$, where T is the number of iterations.

We see that all parameters except Z are constants, which means that the right side of the formula is only affected by Z. However, an increase in Z will cause the right side to increase as a whole. This simultaneously affects the value of the left side of the formula, leading to poorer overall convergence of the model.

3.1.4.6 Aggregate Mesh

The aggregation mesh is the most common in DFL [3,35,45]. Unlike the aggregation star, each device can play the role of the central device and can be trained for local models. Like the Figure 3.6, first of all, each device has preset parameters. Each device is trained locally according to the parameters and then sent to other devices. Each device aggregates parameters separately, and the aggregated parameters are recorded as the initial parameters of the next round. The next round of training is carried out based on the initial parameters. The training process is the same as the aggregate star.

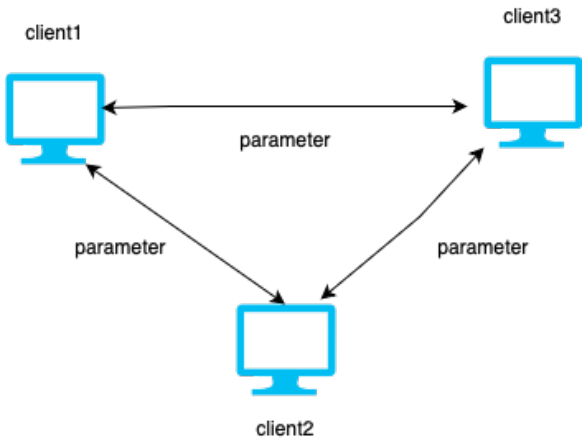

Figure 3.6: Mesh topology.

3.2 Summary and Take Away Message

In this chapter, we first define the network environment and the distributed optimization model for Decentralized Federated Learning (DFL). Since our experiments primarily focus on the impact of topology and aggregation methods on DFL under Non-IID data conditions, we assume that each DFL model operates in the same network environment. Then, we mathematically describe the transmission formulas for six different DFL models, which reflect the specific mechanisms of data transmission and parameter updates for each model.

Next, we conduct an in-depth analysis of the convergence performance of these six DFL models under Non-IID data conditions. We found that as the degree of Non-IID increases, the actual loss values obtained by the six different DFL models diverge more from the ideal loss values. This indicates that Non-IID data distribution significantly affects the convergence of DFL models. Our findings clearly show that the heterogeneity of data distribution is a critical factor influencing the performance of DFL models, which needs to be considered in practical applications.

Our mathematical proof cannot compare the advantages and disadvantages of different DFLs because, in the mathematical proof, metrics such as T (number of rounds) are present in the star topology but not in the linear and ring topologies. Therefore, we cannot directly compare the differences between each DFL (topology, training strategy) using mathematical formulas. Instead, we compared these metrics

through experiments 4.

Chapter 4

Performance Evaluation

This chapter presents the experimental evaluations of the chosen six DFL deployments for traditional, deep neural networks, and Large Language Models (LLMs) with a various degree of non-IID data distribution to illustrate their convergence rates. Thus, users can select appropriate deployment criteria for their targeted applications. We first present the evaluation workflow, setup, implementation, model and data selections. Then, the chapter presents the evaluation results with discussions.

Evaluation Workflow: This section presents the evaluation workflow (see Figure 4.1). First, we select the DFL deployment topology and the aggregation approach, which is six in our case. Next, we select the learning models we will evaluate along with the corresponding datasets. After that, we must define the non-IID data distribution for the chosen model and data. Finally, we perform the distributed training and test the model performance. In the following, we present details of the workflow. Note that we do not present the topology and training schemes in the following, which is introduced in Section 2.1.4 and Section 2.1.5.

4.1 Model and Data Selection

To ensure that our research adapts to multiple fields, we have chosen three different types of learning models: traditional, deep learning, and large language models. The loss functions of traditional models are often convex, while those for deep learning and large language models are nearly convex. The models and their datsets are listed in Table 4.1.

Traditional models: We selected Support Vector Machines [17] (SVM) and Logistic Regression [20] to be tested over the *Breast Cancer Dataset* [38]. First, both models are suitable for various data types and tasks, including classification and regression problems, making them highly applicable and flexible. Second, SVM and regression models have a solid theoretical foundation and extensive research literature

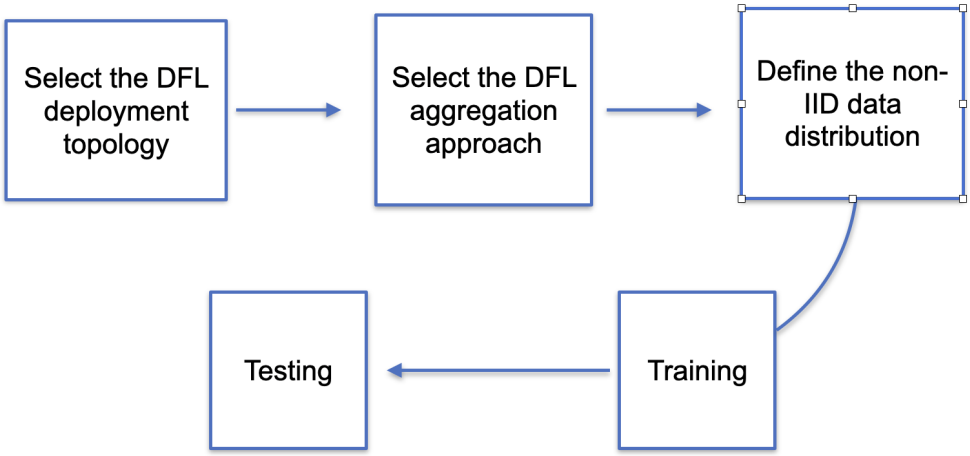

Figure 4.1: Evaluation workflow.

support. Their algorithmic complexity and performance have been thoroughly studied and validated. Additionally, logistic regression is chosen because it is frequently used in other federated learning (FL) papers [18, 23], demonstrating its effectiveness and reliability in distributed environments.

We used Stochastic Gradient Descent (SGD) [6] as the optimizer for these traditional models and conducted incremental training while recording the loss values at each epoch. This method not only improves the model's performance on non-independent and identically distributed data but also enhances its adaptability in a distributed learning environment.

Deep learning: For deep learning, we selected ResNet [15] for vision and DistilBERT [7] for NLP tasks. ResNet [15], short for Residual Network, is a type of convolutional neural network that uses on computer version. DistilBERT (Bidirectional Encoder Representations from Transformers) is profoundly effective for natural language processing tasks. Similarly, we employed SGD as the optimizer for deep learning models and conducted distributed training. The benefits of this approach include better control over the convergence and final outcomes of deep learning models, enhanced generalization capabilities, increased training efficiency, reduced risk of overfitting, improved model stability and robustness, and flexible handling of large-scale datasets.

We chose ResNet because of its outstanding performance in image recognition

tasks, its strong theoretical foundation, and its widespread application support. ResNet effectively addresses the vanishing gradient problem in deep neural networks through the introduction of residual connections, enabling the training of deeper networks.

Specifically, we used the ResNet-18 version. This choice was made because ResNet-18 has a relatively shallow structure, low computational cost, and is suitable for experiments in resource-constrained environments, while still providing strong image feature extraction capabilities.

We used MNIST dataset [43] for ResNet model. The dataset consists of a large collections of handwritten digits, is commonly used in the field of image processing to train models to recognize numbers effectively, making it ideal for demonstrating ResNet's capabilities in image classification.

Large Language Models (LLM): For LLMs, we chose MiniGPT-4 [48] as our model. MiniGPT is based on the GPT (Generative Pre-trained Transformer) architecture, a vision-language model that can take an image from a client and then answer questions about it. Similarly, we used SGD as the optimizer and conducted distributed training. However, unlike traditional and deep learning models, due to the enormous size of MiniGPT-4, its training is divided into two parts: the first part trains the connections between the vision and language models, and the second part involves fine-tuning the model. In this experiment, we only trained the fine-tuning part.

We used *cc sub dataset* [48], a text-image dataset designed to enhance language model training with multi modal inputs. The *TREC_6 dataset* [8], known for its use in evaluating text classification and information retrieval systems, has been selected for training the BERT model. This dataset comprises a set of questions categorized into six different types, challenging the model to understand and process natural language queries accurately.

Table 4.1: Model Dataset Table

Model	**Dataset**
SVM	Breast Cancer Dataset
Logistic Regression	Breast Cancer Dataset
ResNet-18	MNIST dataset
DistilBERT	TREC_6 dataset
MiniGPT-4	cc sub dataset

4.2 Non-IID Data Distribution

In our experiment, we chose to focus on the most prevalent form of NON-IID data—label size imbalance—as this is commonly observed in vertical Federated Learning models [49]. Label imbalance is defined as the variation in the distribution of dataset labels across different clients. In the following, we present the process of generating multiple levels using two approaches presented in [16].

Non-IID Data Levels for Traditional Models: Given that SVM and logistic regression are binary classifiers and use the Breast Cancer Dataset, which contains only two labels, we employed three different levels of non-IID data. To determine these levels, we used KL divergence as the metric [16].

First, let us explain what KL divergence is. The Kullback-Leibler (KL) divergence between two probability distributions P and Q is defined as:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{4.1}$$

where $P(i)$ is the probability of event i in distribution P, and $Q(i)$ is the probability of the same event in distribution Q. KL divergence measures the difference between the distributions of two datasets. The greater the difference in the distributions, the higher the KL divergence value. For example, if the first dataset is $[1, 1, 1, 1, 1]$ and the second dataset is also $[1, 1, 1, 1, 1]$, then the distributions are identical, and the KL divergence value is 0.

We represent the distribution of positive labels on five devices as an array, while the distribution of negative labels is represented as "1 - positive label values." For instance, if the distribution of positive labels on the five devices is $[1, 1, 1, 1, 1]$, the distribution of negative labels would be $[0, 0, 0, 0, 0]$; if the distribution of positive labels is $[0.1, 0.3, 0.5, 0.7, 0.9]$, then the distribution of negative labels would be $[0.9, 0.7, 0.5, 0.3, 0.1]$. We then filtered the dataset on each device, using 10% of the positive labels and 90% of the negative labels from the complete dataset on the first device, and 30% of the positive labels and 70% of the negative labels on the second device.

Finally, we divided the non-IID distribution of the data into three groups using KL divergence. Specifically, we established three different distributions on five devices: $[1, 0, 0.7, 1, 0]$, $[0.5, 0.6, 0.7, 0.8, 0.9]$, and $[0.1, 0.3, 0.5, 0.7, 0.9]$. These values represent

the proportion of positive labels in each device's database. We compared these distributions with the complete database's distribution (all $[1, 1, 1, 1, 1]$) and calculated the KL divergence, obtaining values of 0.52371, 0.0206, and 0.18013, which represent three different levels. This method ensures that each device has all labels, but the label distribution varies between devices.

Non-IID Data Levels for Deep Learning Models: We follow the leveling approach presented in [16] for multi-label data used in deep learning, i.e., consider two scenarios: 1) each device has all labels but the label distribution differs between devices and 2) each device does not have all labels and the label distribution differs between devices as well. In total, we have five different levels of Non-IID settings in our experiment. Levels 1 and 2 are consistent with previous methods, where we use KL divergence as a reference to verify the performance of DFL (Decentralized Federated Learning) when each device has access to all labels. From Level 3 to Level 5, these settings were designed to assess the performance when each device does not have access to all labels. Specifically, devices in Level 3 have 90% of the labels from the complete dataset, Level 4 has 70% of the labels, and Level 5 has 50% of the labels.

Specifically, in the first scenario, we also used KL divergence. For example, suppose $[0.1, 0.3, 0.5, 0.7, 0.9]$ represents the distribution of odd-numbered labels on five devices, while the distribution of even-numbered labels is $[0.9, 0.7, 0.5, 0.3, 0.1]$. We compared the distribution of odd-numbered labels with the complete dataset distribution $[1, 1, 1, 1, 1]$ and calculated the KL divergence. For this scenario, we selected two levels, $[0.5, 0.6, 0.7, 0.8, 0.9]$ and $[0.1, 0.3, 0.5, 0.7, 0.9]$. Odd-numbered labels use the original distribution, while even-numbered labels use the values obtained by subtracting from 1. Their KL divergence values are 0.0206 and 0.18013, respectively.

In the second configuration, we randomly selected 90%, 70%, and 50% of the labels from the complete dataset to form the dataset for each local model. For example, the MNIST dataset contains ten different labels; for each device, we randomly selected 9, 7, and 5 labels, corresponding to three different levels. Each selection was performed randomly to ensure the uniqueness of the data distribution on each device.

Non-IID Data Levels for Large Language Models: Regarding large language models, considering that the cc_sub dataset is inherently a non-IID dataset composed of images and text, we decided not to further stratify it. In this dataset,

each image and its associated text differ from others, naturally exhibiting non-IID characteristics [48].

4.3 Experimental Setup and Implementation

4.3.1 Setup

In our experiments, we used the NVIDIA Quadro RTX 8000 GPU. This device is equipped with 48GB of memory, driver version 535.86.05, and CUDA version 12.2. We simulated five different devices on this equipment, each capable of training models and sending parameters to each other.

4.3.2 Implementation

We implemented the DFL framework using Python 3.12.1, with NumPy version 1.26.4 and scikit-learn version 1.5.0. The local model was programmed using PyTorch version 2.3.0+cu121. Additionally, we set up MiniGPT-4 following the guidelines provided in [48].

4.3.2.1 Model Implementation and Training

In order to control variables and accurately study the impact of different topologies, data distributions, and convergence methods on Decentralized Federated Learning (DFL), we ensured that each local model used the same hyper-parameter settings as the baseline model. This consistency in hyper-parameters was maintained regardless of whether the local model was a traditional model, a deep learning model, or a large language model. By keeping the hyper-parameter settings uniform, we aimed to isolate the effects of the topologies, data distributions, and convergence methods on the performance of DFL.

This approach ensures that in different DFL configurations, each model not only has the same architecture but also uses the same gradient descent strategy for training. By unifying the hyperparameters, we can more accurately evaluate the impact of different topologies on model performance, rather than performance variations caused by differences in hyperparameters. This method allows us to effectively isolate and analyze how different network structures and data strategies affect model learning

efficiency and convergence speed, thereby providing valuable insights for model optimization in decentralized learning environments.

SVM: Specifically, for the SVM model, we selected 1000 as the L2 regularization strength, which helps our loss function be strongly convex. During feature selection, we used a correlation threshold (`corr_threshold`) of 0.9 to remove highly correlated features, while setting the significance level (`sl`) to 0.05 to remove features with a significance lower than this threshold. In terms of model training, we set the learning rate (`learning_rate`) to 0.00001, the maximum epochs (`max_epochs`) to 500, and the batch size to 1.

To evaluate model performance, we used 80% of the data for training and 20% for testing. Additionally, during data preprocessing, we used different percentages to create imbalanced datasets to test the model's performance on imbalanced datasets. For the baseline, at 500 epochs, we found that the loss no longer changes and stabilizes around 595.7978 (using 10000 as the regularization strength). Without L2 regularization, the original loss value is 0.102, and the F1 score is 0.9545.

Logistic Regression: In the provided code, several hyper-parameters are set to control the training process of the machine learning models. The `regularization strength` is set to 10000, which helps in preventing over-fitting by penalizing large weights in the model. The `learning_rate` is set to 0.00001, which determines the step size at each iteration while moving towards a minimum of the loss function. During feature selection, a correlation threshold (`corr_threshold`) of 0.9 is used to remove highly correlated features, and a significance level (`sl`) of 0.05 is employed to discard less significant features during the training process.

For training the models, the number of epochs (`max_epochs`) is set to 500, indicating the total number of times the learning algorithm will work through the entire training dataset. Additionally, the batch size is set to 1, meaning the model will update its parameters after every single training example. The dataset is split into training and testing sets, with 80% of the data used for training and 20% for testing. During data preprocessing, the data is normalized using the `MinMaxScaler`, and different percentages of the data are used to create imbalanced datasets to test the model's performance under varying conditions.

ResNet18: For image classification tasks, we implemented the ResNet18 model.

The learning rate (`learning_rate`) was set to 0.01 to manage the gradient descent step size, the batch size (`batch_size`) was established at 64 to control the data volume processed per iteration, and the number of epochs (`max_epochs`) was set to 100 to define the maximum training iterations.

The MNIST dataset was utilized, and label size imbalance was introduced to evaluate the model's performance. Throughout training and evaluation, we applied the Cross-Entropy Loss function to compute losses and used accuracy, F1 score, and recall as evaluation metrics for a comprehensive assessment of the model's performance. As the epoch count neared 100, the loss parameter stabilized, reaching approximately 0.0039 on the validation set with an F1 score of 0.9893 by the hundredth epoch.

The MNIST dataset was utilized, and label size imbalance was introduced to evaluate the model's performance. We split the dataset into 80% for training, 10% for validation, and 10% for testing. Throughout training and evaluation, we applied the Cross-Entropy Loss function to compute losses and used accuracy, F1 score, and recall as evaluation metrics for a comprehensive assessment of the model's performance. As the epoch count neared 100, the loss parameter stabilized, reaching approximately 0.0039 on the validation set with an F1 score of 0.9893 by the hundredth epoch.

DistilBERT: We utilized DistilBERT as the base model. The hyperparameters were set as follows: the learning rate (`learning_rate`) was fixed at 1e-5 to control the step size of gradient descent, the batch size (`batch_size`) was set to 16 to determine the amount of data processed in each iteration, and the number of epochs (`max_epochs`) was set to 20 to limit the maximum iterations for training the model.

We partitioned the TREC-6 dataset into training and testing sets, with a test size of 20%. During the model training and evaluation phases, we employed the SGD optimizer to update model parameters and used accuracy, F1 score, and recall as evaluation metrics to comprehensively assess the model's performance. As the number of epochs approached 20, we observed that the loss parameter ceased to change, stabilizing at approximately 0.1961 on the validation set with an F1 score of 0.9733 by the twentieth epoch.

MiniGPT-4: In the case of MiniGPT-4, we performed image-text pre-training based on the pre-trained LLaMA2. The model's maximum text length was set to 160, with `"</s>"` as the end-of-sequence token, and the prompt template was `"[INST]`

[/INST]". The cc_sbu_align dataset was employed, with a batch size (batch_size) of 12 and an image size (image_size) of 224.

Training utilized a linear warmup cosine learning rate schedule, with an initial learning rate (initial_learning_rate) of 3e-5, a minimum learning rate (min_learning_rate) of 1e-5, a warmup learning rate (warmup_learning_rate) of 1e-6, and a weight decay (weight_decay) of 0.05. The maximum number of training epochs (max_epochs) was set to 5, with each epoch iterating 200 times and a warmup step count (warmup_steps) of 200. We enabled automatic mixed precision (AMP) and distributed training using CUDA devices, with a random seed (random_seed) set to 42 and logging enabled via Weights and Biases. These parameters were chosen to align with MiniGPT-4's default settings, ensuring consistency with the original implementation.

4.4 Evaluation metrics and comparison

We have defined a baseline model, referred to as F^*, which represents the state with the minimum loss value across the entire model. Mathematical proofs have confirmed that F^* constitutes the optimal solution under ideal conditions. Such ideal conditions presume the existence of a machine with unlimited computational power and memory, capable of processing the entire dataset. In essence, the baseline can be regarded as the optimal model outcome obtained under ideal circumstances.

However, in the practical application of Decentralized Federated Learning (DFL), to protect user privacy, each participating device can only access its own subset of data, and the computational capacity of each device is limited. Therefore, by comparing different DFL models with this ideal baseline, we can measure the gap between each model and the ideal optimal solution, thus evaluating their effectiveness and efficiency in practical applications.

The first experiment aimed to verify whether six different Decentralized Federated Learning (DFL) models could all achieve convergence. To ensure the accuracy of the experiment, we took the following measures: each device had a complete 100% dataset and used the same hyperparameter configuration as the baseline model. This setup ensured that each participating device started the experiment from the same point, eliminating any variance factors that could arise from uneven data distribution or hyperparameter differences. By this method, we could directly observe and evaluate

the convergence performance of each DFL model in a completely homogeneous data environment.

The second experiment aimed to verify whether different Decentralized Federated Learning (DFL) models would be affected by varying degrees of non-IID data. To ensure the accuracy of the experiment, we took the following measures: as described above, we selected different degrees of non-IID datasets and used the same hyper-parameter configuration as the baseline model. This method allowed us to directly observe and evaluate the convergence performance of each DFL model in a non-IID data environment.

We evaluated the convergence of different DFL models by examining the loss curves. Specifically, when the loss curve becomes gradually flatter or when the training loss and validation loss curves intersect, we consider the model converged.

Loss: The loss function used was Cross-Entropy Loss, optimized using Stochastic Gradient Descent (SGD) - as defined below. The Cross-Entropy Loss is given by

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right].$$

SGD was used for optimization due to its simplicity, ease of implementation, and suitability for large-scale datasets. It updates the parameters in each iteration, speeding up the model training process.

F1 Score: We used binary classification for traditional machine learning datasets and multi-class classification for deep learning datasets. Thus, we calculated both the binary classification F1 score and the multi-class classification F1 score. The binary classification F1 score is given by

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

while the multi-class F1 score is calculated using macro-averaging as

$$F1_{\text{macro}} = \frac{1}{N}\sum_{i=1}^{N} F1_i.$$

This approach comprehensively measures the model's performance across different classes. For multi-class tasks, the macro-averaged F1 score reflects the balanced performance of each class, avoiding bias towards classes with larger data volumes.

LLM Accuracy Testing: To test the LLM accuracy as described in [48], we will ask the model four different questions in the form of multiple-choice questions:

- Help me draft a professional advertisement for this.
- Can you craft a beautiful poem about this image?
- Explain why this meme is funny.
- How should I make something like this?

We will provide the model with an image and then ask these questions to see if there are any obvious errors in the responses, such as mentioning objects that are not present in the image. If there are obvious errors, we will give this task a score of 0. If there are no obvious errors, we will give this task a score of 1.

4.5 Baseline results

This section presents the performance of all five models with the baseline setup. We defined the baseline model, F^*, as the state with the minimum loss value in the overall model. Through mathematical proof, we have determined that F^* represents the optimal solution under ideal conditions. Such an ideal condition assumes the existence of a machine with unlimited computational power and memory, capable of handling the entire dataset. In short, the baseline can be seen as the optimal model outcome under ideal circumstances.

However, in practical applications of Decentralized Federated Learning (DFL), to protect user privacy, each participating device can only access its own data, and each device has limited computational capabilities. Therefore, by comparing different DFL models with this ideal baseline, we measure the gap between actual and the ideal solution, thereby evaluating their effectiveness and efficiency in real-world applications.

4.5.1 SVM

The Figure 4.2 shows the convergence graph of our SVM. We can observe that the curves for training loss and validation loss are very smooth and continuously decreasing. When the epoch approaches 400 to 500, the loss stabilizes and remains almost

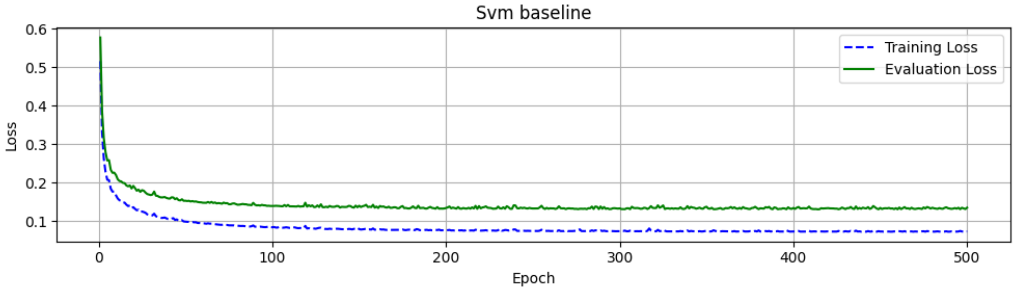

Figure 4.2: SVM baseline.

unchanged. If, during training, the training loss continues to decrease while the validation loss starts to increase after reaching a certain point, this is usually a sign of over-fitting. However, this phenomenon does not appear in our graph. Although the training set and the validation set do not overlap, the gap between them is reasonable, only 0.04. The accuracy and F1 score on the test dataset are 0.9737 and 0.9663, respectively.

4.5.2 Logical Regression

The Figure 4.3 shows the convergence graph of our logical regression model. We can see that the curves for training loss and validation loss are very smooth and continuously decreasing. When the epoch approaches 800 to 1000, the loss stabilizes and remains almost unchanged. We observe that the training loss and validation loss curves are very close to each other and follow the same trend, indicating that the model does not exhibit over-fitting. Although the training loss and evaluation loss do not completely overlap, the gap between them is small and acceptable, only 0.03. The accuracy and F1 score on the test dataset are 0.9561 and 0.9438(It is very great phenomenon), respectively.

4.5.3 ResNet

The Figure 4.4 show the convergence for our resnet model. As we know, resnet model is not convex model. Thus, by adjusting the hyper-parameters, we trained the MNIST dataset using ResNet-18 and recorded the training loss and evaluation loss for each training epoch. We observed that after a sufficient number of training epochs (100

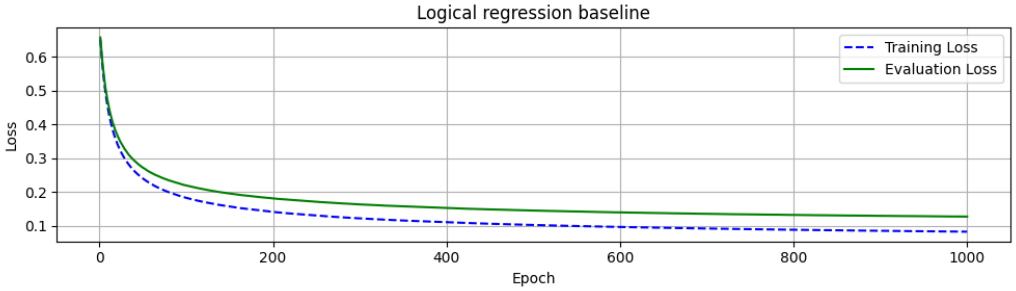

Figure 4.3: Logical regression baseline.

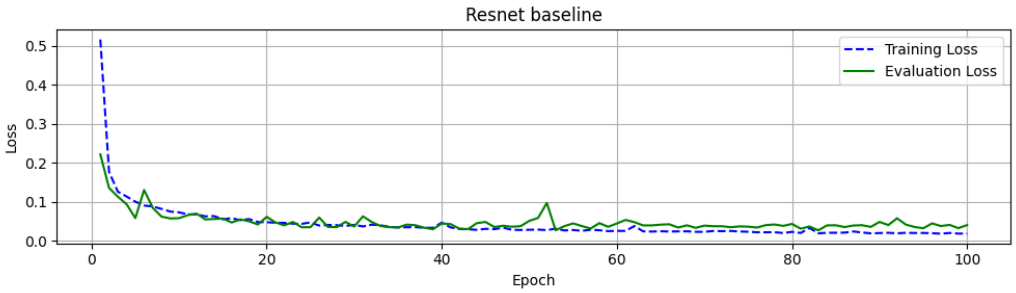

Figure 4.4: Resnet baseline.

epochs), both the training loss and evaluation loss approached zero (training: 0.01, evaluation: 0.03). The F1 score and accuracy on the test dataset were both close to 1 (0.9895 and 0.9893, respectively). There was no sign of over-fitting, although the evaluation loss showed some fluctuations, but they were not significant. Furthermore, there were substantial periods where the two losses overlapped, indicating that the model has similar performance on both the training set and the evaluation set.

4.5.4 DistilBERT

Figure 4.5 shows the convergence for our DistilBERT model. Same with Resnet model, the DistilBERT model is also not convex model. Thus, by adjusting the hyper-parameters, we trained TREC 6 dataset using the DistilBERT model and recorded the training loss and evaluation loss for each training epoch. We observed that after a sufficient number of training epochs (17 epochs), the training loss approached zero, the evaluation loss approached to 0.1, and they both remain almost unchanged. There was no sign of over-fitting, although the evaluation loss showed some fluctuations, but

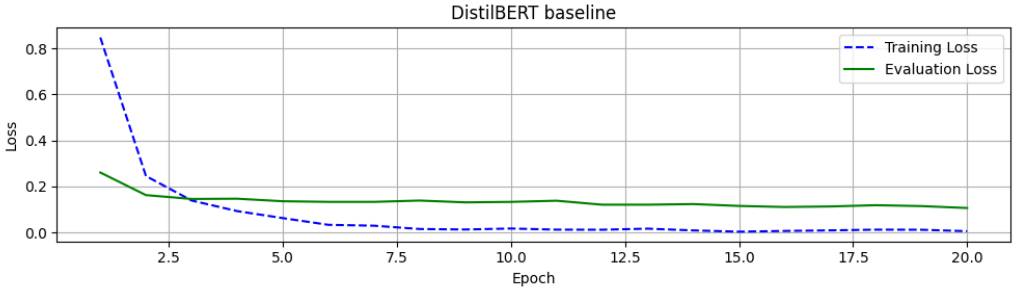

Figure 4.5: DistilBERT baseline.

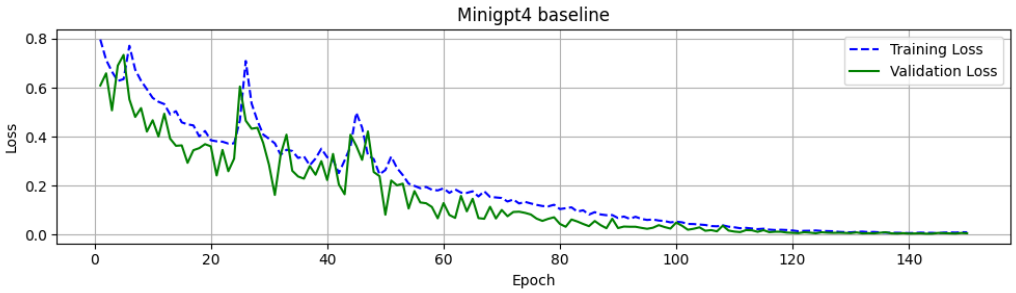

Figure 4.6: Mini-GPT baseline.

they were not significant. The accuracy and F1 score on the test dataset are 0.9680 and 0.9733, respectively.

4.5.5 Mini-GPT

Figure 4.6 shows the convergence graph of our Mini-GPT's baseline model. We can see that the curves for training loss and validation loss are continuously decreasing. When the epoch approaches 100 to 120, the loss stabilizes and remains almost unchanged. We observe that the training loss and validation loss curves are very close to each other and follow the same trend, indicating that the model does not exhibit overfitting. Although the training loss and evaluation loss do not completely overlap, the gap between them is small and acceptable. Since Mini-GPT needs to be measured manually, we do not provide accuracy and F1 values here. We asked five groups of questions for each device, four questions for each group, and then observed whether there were any questions that would go wrong. We found that 18 of the 20 questions were correct and 2 were wrong. So, the accuracy of the baseline is 0.9.

4.6 Impact of Topologies on the Convergence Rate

In this section, we will use two different types of results (F1 score and convergence rate) to demonstrate the performance of different DFL models. For linear and ring topologies, the F1 score is the average of the five devices. For star and mesh topologies, it is the average of multiple rounds. The coverage rate shows the epoch at which each client converges, providing a more comprehensive analysis of the DFL convergence performance. This not only allows us to verify whether all six DFL models converge under IID data conditions, consistent with our mathematical analysis, but also to investigate how the six DFL models are affected by Non-IID data.

First, it is essential to ensure that the total number of epochs used in all DFL configurations is consistent with the baseline. For example, if the baseline SVM model is trained for a total of 500 epochs, then all six DFL configurations using SVM as the local model should also be trained for a total of 500 epochs. Secondly, we aim to ensure that each device operates with the same number of epochs. Continuing with the SVM example, if there are five devices, each device would run for 100 epochs. This fixed-epoch strategy helps maintain consistency and fairness across different experiments. By standardizing the number of epochs, we can more accurately compare the performance of different DFL structures under the same conditions.

Before presenting the results, we need to emphasize several points. First, all hyperparameters are kept consistent with the baseline. Second, to ensure the fairness of the experiments, we used the same number of rounds and epochs per client in all evaluations. This setup aims to ensure that our experimental results are reliable and can effectively compare the performance of different topologies and model configurations. We then list the details according to different DFL topologies, which helps us better compare the impact of different topologies. Finally, we will introduce the number of rounds and epochs each client needs to run for each DFL topology.

For the linear topology, since each client only trains once, the number of epochs on each client is equal to the total number of epochs divided by the number of clients. Specifically, according to the baseline, we found that SVM needs to run a total of 500 epochs. With 5 devices, each device runs 100 epochs. Compared to the linear topology, the ring topology has one more training round than the linear topology. Therefore, its number of epochs is halved. In this case, the formula for epochs is:

*epoch = total_epoch / (Num_client * num_Round)*. Specifically, according to the baseline, we found that SVM needs to run a total of 500 epochs. With 5 devices, each device runs 50 epochs and trains for two rounds.

For the star and mesh topologies, since they are similar to concurrent training, the number of epochs on each client for each round is equal to the total number of epochs divided by the number of rounds. For example, according to the baseline, we found that SVM needs to run a total of 500 epochs. We run a total of 5 rounds, so each device runs 100 epochs. The specific details are listed below.

4.6.1 Continues Linear

Table 4.2 shows the convergence results of C_linear across five different local models and the number of epochs run on each device. We found that for logistic regression, the first device did not converge because the model could not converge within 200 epochs, which is consistent with the baseline. For the other local models, each device was able to converge, also inline with the baseline.

Table 4.2: Convergence rate of continuous linear topology.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	83	100	200	300	400	100
Regression	NC	230	400	600	800	200
ResNet	18	27	57	75	87	20
DistilBERT	4	8	11	15	19	4
MiniGPT	22	31	61	91	121	30

We found that the F1 scores in Table 4.3 for all devices are very similar, with no significant differences, and they are close to the baseline. These results underscore the effectiveness of continues linear in maintaining model stability and accuracy.

Table 4.3: F1 score of continuous linear.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**Average F1**
SVM	0.9545	0.9663	0.9663	0.9663	0.9663	0.9639
Logic	0.9176	0.9302	0.9425	0.9425	0.9545	0.9375
ResNet	0.9896	0.9885	0.9857	0.9869	0.9838	0.9869
DistilBERT	0.9176	0.9302	0.9425	0.9425	0.9545	0.9375
MiniGPT	0.8000	0.9000	0.9000	0.9000	0.9000	0.8800

4.6.2 Continues Ring

Table 4.4 above shows the convergence results of C_ring across five different local models and the number of epochs run on each device. Each device was able to converge, which is consistent with the baseline. Additionally, we found that for logistic regression, the first device could converge because, although the model could not converge within 200 epochs, the ring topology allows for multiple rounds of training. Thus, Device 1 was able to converge by the 500th epoch, demonstrating the superiority of the ring topology over the linear topology.

Table 4.4: Convergence rate of continuous ring topology.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	250	80	100	150	200	50
Logic	500	233	300	400	500	100
ResNet	10	19	25	88	99	10
DistilBERT	12	4	16	18	20	2
MiniGPT	76	26	30	45	60	1

Same with continues linear, we found that the F1 scores in Table 4.4 for all devices are very similar, with no significant differences, and they are close to the baseline.These results underscore the effectiveness of continues ring in maintaining model stability and accuracy. We found the f1 on client is greater than continuous linear topology, this highlights the advantages of the ring topology.

Table 4.5: F1 score of continuous ring.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**Average F1**
SVM	0.9645	0.9663	0.9545	0.9545	0.9663	0.9604
Logic	0.9425	0.9545	0.9438	0.9438	0.9438	0.9457
ResNet	0.9817	0.9798	0.9791	0.9831	0.9823	0.9789
DistilBERT	0.9425	0.9545	0.9438	0.9438	0.9438	0.9457
MiniGPT	0.7000	0.8000	0.8000	1.0000	0.9000	0.8400

4.6.3 Aggregate Linear

Table 4.6 shows the convergence results of A_linear across five different local models and the number of epochs run on each device. We found that for logistic regression,

the first device did not converge because the model could not converge within 200 epochs. Observing the baseline curve, we found that this is consistent with the baseline. For the other local models, each device was able to converge. Observing the baseline curve, we found that this is also consistent with the baseline.

Table 4.6: Convergence rate of aggregate linear topology.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	92	100	200	300	400	100
Logic	NA	250	400	600	800	200
ResNet	19	25	55	78	90	20
DistilBERT	3	7	12	15	20	4
MiniGPT	23	31	61	91	121	30

We found that the F1 scores in Table 4.7 for all devices are very similar, with no significant differences, and they are close to the baseline. These results underscore the effectiveness of aggregate linear in maintaining model stability and accuracy.

Table 4.7: F1 score of aggregate linear

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**Average F1**
SVM	0.9425	0.9663	0.9545	0.9663	0.9545	0.9568
Logic	0.9176	0.9302	0.9425	0.9425	0.9545	0.9375
ResNet	0.9851	0.9824	0.9837	0.9854	0.9875	0.9848
DistilBERT	0.9176	0.9302	0.9425	0.9425	0.9545	0.9375
MiniGPT	0.8000	0.8000	0.9000	0.8000	0.9000	0.8400

4.6.4 Aggregate Ring

Table 4.8 above shows the convergence results of A_ring across five different local models and the number of epochs run on each device. Each device was able to converge, which is consistent with the baseline. Additionally, we found that for logistic regression, the first device could converge because, although the model could not converge within 200 epochs, the ring topology allows for multiple rounds of training. Thus, Device 1 was able to converge by the 500th epoch, demonstrating the superiority of the ring topology over the linear topology.

We found that the F1 scores in Table 4.9 for all devices are very similar, with no significant differences, and they are close to the baseline. These results underscore the

Table 4.8: Convergence rate of aggregate ring topology.

MODEL	Client1	Client2	Client3	Client4	Client5
SVM	250	95	100	150	200
Logic	500	250	300	400	500
ResNet	10	18	25	89	98
DistilBERT	11	4	15	18	20
MiniGPT	76	24	30	45	60

effectiveness of aggregate linear in maintaining model stability and accuracy across different conditions. We found the f1 on client is greater than continuous linear topology,this highlights the advantages of the ring topology.

Table 4.9: F1 score of aggregate ring

MODEL	Client1	Client2	Client3	Client4	Client5	Average F1
SVM	0.9545	0.9563	0.9545	0.9545	0.9545	0.9569
Logic	0.9302	0.9425	0.9425	0.9425	0.9425	0.9400
ResNet	0.9806	0.9761	0.9798	0.9827	0.9804	0.9805
DistilBERT	0.9302	0.9425	0.9425	0.9425	0.9425	0.9400
MiniGPT	0.7000	0.8000	0.9000	0.9000	0.9000	0.8400

4.6.5 Star and Mesh

Table 4.10 4.11 above shows the convergence results of Star and Mesh topologies across five different local models and the number of epochs run on each device. Each device was able to converge, which is consistent with the baseline. Additionally, we found that the convergence of Star and Mesh topologies is essentially the same, which also aligns with our baseline.

Table 4.10: Convergence rate of star topology.

MODEL	Client1	Client2	Client3	Client4	Client5
SVM	97	99	93	94	95
Logic	400	401	403	400	402
ResNet	16	25	53	77	92
DistilBERT	4	8	11	15	19
MiniGPT	40	41	40	42	41

We found that the F1 scores in Tables 4.13 and 4.12, all devices are very similar, with no significant differences, and they are close to the baseline. These results

Table 4.11: Convergence rate of mesh topology.

MODEL	Client1	Client2	Client3	Client4	Client5
SVM	96	91	92	98	99
Logic	400	401	402	400	402
ResNet	16	25	53	77	92
DistilBERT	4	8	11	15	19
MiniGPT	40	41	42	40	41

underscore the effectiveness of aggregate linear in maintaining model stability and accuracy. We found the Mesh and star have similar F1 score, this is consistent with their convergence.

Table 4.12: F1 score of star

MODEL	Client1	Client2	Client3	Client4	Client5	Average F1
SVM	0.9545	0.9663	0.9645	0.9645	0.9663	0.9588
Logic	0.9438	0.9438	0.9438	0.9438	0.9438	0.9438
ResNet	0.9820	0.9835	0.9863	0.9871	0.9879	0.9851
DistilBERT	0.9463	0.9291	0.9383	0.9388	0.9352	0.9375
MiniGPT	0.8000	0.9000	0.9000	0.9000	0.9000	0.8800

Table 4.13: F1 score of mesh

MODEL	Client1	Client2	Client3	Client4	Client5	Average F1
SVM	0.9645	0.9663	0.9663	0.9645	0.9663	0.9597
Logic	0.9438	0.9438	0.9438	0.9438	0.9438	0.9438
ResNet	0.9781	0.9881	0.9871	0.9846	0.9843	0.9688
DistilBERT	0.9328	0.9406	0.9222	0.9337	0.9374	0.9333
MiniGPT	0.8000	0.9000	0.9000	0.9000	0.9000	0.8800

4.6.6 Summary and Takeaway

We studied the performance of six DFL models with IID data. We found that whether the local model was a traditional model, a deep learning model, or an LLM, the models were able to achieve convergence. These observations not only validate the effectiveness of our DFL models across different clients but also demonstrate that the models can achieve consistent convergence. This is crucial for designing practical decentralized federated learning systems. Specifically, for traditional models, when

each client has the entire dataset, or in other words, when the data distribution across each device is identical, the DFL model's convergence matches the baseline. This confirms our mathematical derivation that when the Non-IID degree is zero, all models can fully converge, allowing users to achieve the optimal solution.

Notably, for the linear model, we found that device 1 often performed poorly because it did not achieve good convergence when passing parameters to device 2. However, the ring model mitigated this drawback. Additionally, the star and mesh models performed consistently, which aligns with our expectations.

4.7 Impact of Non-IID data Distribution

As in the previous evaluation, we used the same hyper-parameters as the baseline model. It is particularly important to emphasize that, to ensure the fairness of the experiments, we used the same number of rounds and epochs per client in both the previous and this evaluations.

The goal of this evaluation is to investigate the effect of different degrees of Non-IID data on DFL performance. According to our previous definitions, we divided the Non-IID data into different levels. In this section, we will present four different results. The first three shows the convergence results under three different degrees of Non-IID data, detailing the epoch at which each device converges. The last table shows the average F1 scores for different data distributions.

It is important to note that since the LLM dataset is already Non-IID, in this section we only present the performance of DFL using traditional models and deep learning models as local models.

4.7.1 Continues Linear and Aggregate Linear

It is important to note that since SVM and logistic regression use binary classification datasets, they only have three levels of non-IID data. In contrast, for deep learning models, we defined five levels of non-IID data. The first two levels ensure that each device has all labels but with different distributions, whereas the last three levels lack some labels entirely.

Tables 4.14, 4.15, 4.16, 4.18, 4.19, 4.20 show the convergence results for A_linear across five different local models and the number of epochs each device runs. We

Table 4.14: Coverage rate with Level 1 non-IID data distribution for continues linear.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	90	101	NC	NC	NC	100
Logic	NC	399	499	NC	NC	200
ResNet	19	39	59	79	99	20
DistilBERT	4	8	12	16	NC	4

Table 4.15: Coverage rate with Level 2 non-IID data distribution for continues linear.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	NC	260	301	NC	100
Logic	NC	NC	599	799	NC	200
ResNet	19	39	59	79	NC	20
DistilBERT	NC	8	12	16	20	4

Table 4.16: Coverage rate with Level 3 non-IID data distribution for continues linear.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	198	298	NC	NC	100
Logic	NC	NC	599	799	NC	200
ResNet	NC	NC	NC	NC	NC	20
DistilBERT	NC	NC	NC	NC	NC	4

observe that as the level of non-IID data increases, the number of devices that can achieve convergence decreases. However, since SVM and logistic regression do not have data for levels 4 and 5, and because no devices in the continuous linear DFL can converge at levels 3 to 5 when using deep learning models as local models, we only included three tables.

Table 4.17: F1 score of continuous linear.

MODEL	**LEVEL1**	**LEVEL2**	**LEVEL3**	**LEVEL4**	**LEVEL5**
2 SVM	0.88484	0.81312	0.65296	NA	NA
Logic	0.88484	0.74172	0.40426	NA	NA
ResNet	0.97508	0.96920	0.86436	0.58086	0.35598
DistilBERT	0.89014	0.81768	0.95128	0.94374	0.92458

Regarding the F1 score, Tables4.214.3, we notice that except for the DistilBERT model, other models achieve lower F1 scores at higher levels of non-IID data. For the

Table 4.18: Coverage rate with Level 1 non-IID data distribution for aggregate linear.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	89	100	NC	NC	NC	100
Logic	NC	399	499	NC	NC	200
ResNet	19	39	59	79	99	20
DistilBERT	4	8	12	16	NC	4

Table 4.19: Coverage rate with Level 2 non-IID data distribution for aggregate linear.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	197	299	NC	NC	100
Logic	NC	NC	599	799	NC	200
ResNet	19	39	59	79	99	20
DistilBERT	NC	NC	12	16	20	4

Table 4.20: Coverage rate with Level 3 non-IID data distribution for aggregate linear.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	197	299	NC	NC	100
Logic	NC	NC	599	NC	NC	200
ResNet	NC	NC	NC	NC	NC	20
DistilBERT	NC	NC	NC	NC	NC	4

DistilBERT model, there is a noticeable decrease in performance at levels 1 and 2, but surprisingly, level 3 shows higher F1 scores than level 1. This suggests that the DistilBERT model performs better when local devices have incomplete label sets but an equal number of samples for each available label.

Table 4.21: F1 score of aggregate linear.

MODEL	**LEVEL1**	**LEVEL2**	**LEVEL3**	**LEVEL4**	**LEVEL5**
SVM	0.89186	0.79618	0.63902	NA	NA
Logic	0.84936	0.73544	0.52824	NA	NA
ResNet	0.96332	0.97520	0.85048	0.58434	0.34776
DistilBERT	0.86618	0.83402	0.95546	0.94430	0.93760

4.7.2 Continues Ring and Aggregate Ring

As noted earlier, since SVM and logistic regression are based on binary classification datasets, they are only evaluated with three levels of non-IID data. In contrast, deep learning models are assessed with five levels of non-IID data.

Table 4.22: Coverage rate with Level 1 non-IID data distribution for continues ring.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	298	301	350	NC	NC	50
Logic	NC	599	649	NC	NC	100
ResNet	10	20	30	40	50	10
DistilBERT	11	12	14	18	NC	2

Table 4.23: Coverage rate with Level 2 non-IID data distribution for continues ring.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	NC	400	450	NC	50
Logic	NC	NC	799	845	NC	100
ResNet	10	20	30	40	NC	10
DistilBERT	NC	12	14	18	NC	2

Table 4.24: Coverage rate with Level 3 non-IID data distribution for continues ring.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	NC	398	NC	NC	50
Logic	NC	NC	799	NC	NC	100
ResNet	NC	NC	NC	NC	NC	10
DistilBERT	NC	NC	NC	NC	NC	2

Tables 4.22, 4.23, 4.24, 4.25, 4.26, 4.27 show the convergence results for A_linear across five different local models and the number of epochs each device runs. We observe that as the non-IID level increases, fewer devices are able to achieve convergence. However, since there is no data for SVM and logistic regression at levels 4 and 5, and because none of the devices in the continuous linear DFL can converge at levels 3 to 5 when using deep learning models as local models, we have only provided three tables.

Table 4.25: Coverage rate with Level 1 non-IID data distribution for aggregate ring.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	294	298	351	NC	NC	50
Logic	NC	399	499	NC	NC	100
ResNet	10	20	30	40	50	10
DistilBERT	11	12	14	18	NC	2

Table 4.26: Coverage rate with Level 2 non-IID data distribution for aggregate ring.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	NC	400	452	NC	50
Logic	NC	NC	799	845	NC	100
ResNet	10	20	30	40	NC	10
DistilBERT	NC	12	14	18	NC	2

Table 4.27: Coverage rate with Level 3 non-IID data distribution for aggregate ring.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	NC	399	NC	NC	50
Logic	NC	NC	799	NC	NC	100
ResNet	NC	NC	NC	NC	NC	10
DistilBERT	NC	NC	NC	NC	NC	2

Regarding the F1 scores, we observe the same phenomenon as before: all models, except for DistilBERT, achieved lower F1 scores at higher levels of non-IID data. This indicates that the DistilBERT model performs better in situations where local devices do not have all the labels but each label has the same number of samples.

Table 4.28: F1 score of continuous ring.

MODEL	**LEVEL1**	**LEVEL2**	**LEVEL3**	**LEVEL4**	**LEVEL5**
SVM	0.88484	0.81312	0.65296	NA	NA
Logic	0.88484	0.74172	0.40426	NA	NA
ResNet	0.97508	0.96920	0.86436	0.58086	0.35598
DistilBERT	0.74829	0.74444	0.93759	0.86645	0.79190

Table 4.29: F1 score of aggregate ring.

MODEL	LEVEL1	LEVEL2	LEVEL3	LEVEL4	LEVEL5
SVM	0.88614	0.80938	0.64056	NA	NA
Logic	0.85508	0.78278	0.40476	NA	NA
ResNet	0.96342	0.96678	0.84348	0.59818	0.34876
DistilBERT	0.76319	0.74309	0.92209	0.88445	0.78743

4.7.3 Star and Mesh

As with the previous analysis, since SVM and logistic regression are based on binary classification datasets, they are only evaluated with three levels of non-IID data. In contrast, deep learning models are assessed with five levels of non-IID data. Due to the lack of data for levels 4 and 5 for SVM and logistic regression, and because none of the devices in the continuous linear DFL can converge at levels 3 to 5 when using deep learning models as local models, we have only provided three level Tables 4.30, 4.31, 4.32, 4.33, 4.34, 4.35.

Table 4.30: Coverage rate with Level 1 non-IID data distribution for star.

MODEL	Client1	Client2	Client3	Client4	Client5	num_epoch device
SVM	400	402	401	402	401	100
Logic	880	877	878	878	878	200
ResNet	24	24	24	24	24	20
DistilBERT	19	19	19	19	19	4

Table 4.31: Coverage rate with Level 2 non-IID data distribution for star.

MODEL	Client1	Client2	Client3	Client4	Client5	num_epoch device
SVM	400	402	401	402	401	100
Logic	851	852	850	851	852	200
ResNet	24	24	24	24	24	20
DistilBERT	20	20	20	20	20	4

For the F1 scores, the observe phenomena are consistent with earlier findings. This indicates that the DistilBERT model performs better when local devices do not have all the labels but each label contains the same number of samples, a trend that appears across all six DFLs.

Table 4.32: Coverage rate with Level 3 non-IID data distribution for star.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	NC	NC	NC	NC	100
Logic	NC	NC	NC	NC	NC	200
ResNet	NC	NC	NC	NC	NC	20
DistilBERT	NC	NC	NC	NC	NC	4

Table 4.33: Coverage rate with Level 1 non-IID data distribution for mesh.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	401	402	401	402	401	100
Logic	803	802	803	802	804	200
ResNet	24	24	24	24	24	20
DistilBERT	19	19	19	19	19	4

Table 4.34: Coverage rate with Level 2 non-IID data distribution for mesh.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	403	400	401	402	401	100
Logic	830	831	832	831	831	200
ResNet	24	24	24	24	24	20
DistilBERT	20	20	20	20	20	4

Table 4.35: Coverage rate with Level 3 non-IID data distribution for mesh.

MODEL	**Client1**	**Client2**	**Client3**	**Client4**	**Client5**	**num_epoch device**
SVM	NC	NC	NC	NC	NC	100
Logic	NC	NC	NC	NC	NC	200
ResNet	NC	NC	NC	NC	NC	20
DistilBERT	NC	NC	NC	NC	NC	4

Table 4.36: F1 score of star.

MODEL	**LEVEL1**	**LEVEL2**	**LEVEL3**	**LEVEL4**	**LEVEL5**
SVM	0.89398	0.80642	0.65004	NA	NA
Logic	0.85630	0.78278	0.40600	NA	NA
ResNet	0.96658	0.97358	0.84758	0.58502	0.34520
DistilBERT	0.730224	0.72986	0.924468	0.825416	0.785068

Table 4.37: F1 score of mesh.

MODEL	LEVEL1	LEVEL2	LEVEL3	LEVEL4	LEVEL5
SVM	0.88628	0.79840	0.64638	NA	NA
Logic	0.85614	0.78342	0.39276	NA	NA
ResNet	0.97810	0.96286	0.83404	0.58500	0.35000
DistilBERT	0.726764	0.725044	0.913688	0.823308	0.787292

4.7.4 Summary and Takeaway

Classical Models: At level 1 and level 2, star and mesh perform the best. This indicates that star and mesh are more suitable than other topologies for use when the Non-IID level is low. However, at level 3, ring and linear perform significantly better than star and mesh. This suggests that star and mesh do not perform well when the Non-IID level is too high.

The convergence performance of star and mesh is identical, which is consistent with our mathematical analysis. As the Non-IID level increases, convergence performance deteriorates, and the F1 score gradually decreases.

Network topology has a greater impact on device convergence than aggregation strategy. We found that the convergence performance is roughly the same for the same network topology, which is also consistent with our mathematical analysis. The linear topology usually performs poorly on the first device, while the ring topology can compensate for this shortcoming.

Deep Learning Models: Models cannot converge at level 3 or higher Non-IID levels, indicating that no network topology can adapt to the scenario where devices lack full label sets. This suggests that in real-world applications, devices with incomplete label sets should be filtered out, as they would affect the convergence of the global model. When devices have complete label sets, star and mesh perform better under uneven label distributions. Users should prefer star or mesh topologies.

Additionally, for both deep learning and traditional models, we observe that the more evenly distributed the data on each device, the better the convergence and the higher the F1 score. For individual devices, if the distribution of labels within the device is more balanced (e.g., equal distribution of odd and even labels with the same number of samples), the device's performance will be better.

In summary, our conclusions are extendable, particularly when future users wish

to combine different topologies, such as integrating linear and mesh structures. They can leverage our experimental results to select the topology and aggregation method that offer better convergence under various data distributions. For instance, we found that star and mesh topologies perform best when the degree of non-IID is relatively low. Therefore, users might consider using star or mesh as the selected topology in certain scenarios.

The experimental results demonstrate that star and mesh topologies exhibit the best performance under low non-IID conditions. We believe this is because, in a star topology, each device communicates directly with a central device, allowing the central device to more effectively average the parameters from different devices, thereby enhancing convergence. Similarly, in a mesh topology, where each device is interconnected with others, each device can obtain parameters from various devices, resulting in better overall performance.

Chapter 5

Future Work and Conclusion

5.1 Future Work

Although our project has made some progress in analyzing the the convergence of decentralized federated learning (DFL), there are still many directions worth further investigation. Here are some suggestions for future work.

Impact of Heterogeneous Devices: The current research assumes that all devices have the same communication and computing capabilities. However, in practical applications, different devices may have different capabilities. For example, some devices may have extremely high computing power, high memory, and good communication capabilities. Future research could consider the impact of device heterogeneity and analyze how different device capabilities affect the performance and convergence of DFL.

Asynchronous/Synchronous Training: In addition to the three influencing factors, we believe that training synchronously or asynchronously can also impact the convergence of DFL. Compared to CFL, DFL offers more flexibility in training options, allowing users to decide whether to use synchronous or asynchronous training modes based on their specific needs or the computational and transmission capabilities of different devices. This can help reduce the overall time required for the model. We plan to explore this point in future work.

Real System Deployment: We evaluated the performance in a simulation environment to measure the correctness of our analysis. But in order to measure real performance of these DFL deployments, we need to consider real system deployment. For instance, we can deploy the chosen models over five clients machines forming six different topologies. The can be further be extended to the software-defined networking (SDN) paradigm [10] to have agility and scale in meeting dynamically changing demands from applications.

Additionally, our study assumes reliable and lossless communication channels, but

in real-world applications, communication channels may be subject to interference and packet loss. Also, clients can fail or be idle. Future research could introduce models of unreliable communication channels and analyze the impact of packet loss rates and communication delays on DFL performance. This would provide a better investigation into the energy consumption, computing power, and operational speed of DFL. SDN can also be leveraged in designing reliable solutions as pointed in [11–14, 19, 21, 28, 32, 36, 39, 40]. Also, we plan to investigate the impact of failed or idles nodes on the convergence rate. For example, we found that devices with a higher degree of non-IID data tend to degrade overall convergence performance, while devices with more balanced data generally perform better. Based on this observation, one could decide whether to discard or wait for a device to be repaired, balancing overall efficiency with total training time. Similarly, future research could also consider idle devices.

Scalability: We demonstrated the performance of the chosen DFL deployments over five clients. The results confirmed the correctness of the mathematical analysis. We expect that the performance trend will be similar with an increasing number of clients following the same analysis. However, we need extensive performance evaluation with a large number of clients to check how various DFL deployment can scale.

5.2 Conclusion

The aim of this thesis is to investigate the impact of network topology, Non-IID data, and training strategies on the convergence of Decentralized Federated Learning (DFL). We also examined the performance of different DFL deployments on test datasets under varying degrees of Non-IID data. Specifically, we conducted both mathematical and experimental analyses to comprehensively study these factors.

Firstly, we derived mathematical expressions for six practical DFL deployments and defined the optimization objective functions for DFL. Using convex optimization, we analyzed the convergence of each DFL model and found that when data distribution across devices is IID, the difference between the ideal and actual solutions is a constant. As the degree of Non-IID data increases, this difference becomes larger. We also discovered that the convergence analysis for star and mesh topologies is identical.

Next, we conducted two sets of experiments to analyze the convergence of DFL models. We used three different local models: traditional machine learning models, deep learning models, and large language models, and established their baselines. First, we considered IID data on each device and measured the convergence of six deployments. We found that every DFL model could converge regardless of the local model used, consistent with our mathematical analysis. In the second experiments, we used data with varying degrees of Non-IID, we observed that as the Non-IID level increased the convergence performance deteriorated, which also aligned with our mathematical analysis.

Both traditional and deep learning models showed that star and mesh topologies consistently performed well. However, as the Non-IID level increased, convergence performance worsened and F1 scores decreased. For traditional models, star and mesh topologies performed the best under low Non-IID conditions but failed to converge under high Non-IID conditions, performing worse than other topologies. Linear topology generally performed poorly on the first device, while the ring topology compensated for this shortcoming.

For deep learning models, no network topology could support convergence to scenarios where devices had incomplete labels. This implies that in practical applications, devices with incomplete labels should be filtered out as they would affect the global model's convergence. When devices had complete labels, star and mesh topologies performed better under uneven label distributions. Users should prefer star or mesh topologies. Moreover, we found that the more evenly distributed the data on each device, the better the convergence and higher the F1 scores. For individual devices, if the label distribution is more balanced (e.g., equal distribution of odd and even labels with the same number of samples), the device's performance improves.

Bibliography

[1] Ali Al-Shuwaili and Osvaldo Simeone. Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. *IEEE Wireless Communications Letters*, 6(3):398–401, 2017.

[2] Vaishak Belle and Ioannis Papantonis. Principles and practice of explainable machine learning. *Frontiers in big Data*, page 39, 2021.

[3] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*, 2023.

[4] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] Shuzhen Chen, Dongxiao Yu, Yifei Zou, Jiguo Yu, and Xiuzhen Cheng. Decentralized wireless federated learning with differential privacy. *IEEE Transactions on Industrial Informatics*, 18(9):6273–6282, 2022.

[6] J Michael Cherry, Caroline Adler, Catherine Ball, Stephen A Chervitz, Selina S Dwight, Erich T Hester, Yankai Jia, Gail Juvik, TaiYun Roe, Mark Schroeder, et al. Sgd: Saccharomyces genome database. *Nucleic acids research*, 26(1):73–79, 1998.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. Trec complex answer retrieval overview. In *TREC*, 2017.

[9] Hongchang Gao, My T Thai, and Jie Wu. When decentralized optimization meets federated learning. *IEEE Network*, 2023.

[10] I. Haque and N. Abu-Ghazaleh. Wireless software defined networking: a survey and taxonomy. *IEEE Communications Surveys and Tutorials*, 18(4):2713–2737, May 2016.

[11] Israat Haque, Saiful Islam, and Janelle Harms. On selecting a reliable topology in wireless sensor networks. In *Proceedings of the 2015 IEEE International Conference on Communications*, ICC '15, 2015.

[12] Israat Haque and M. A. Moyeen. Revive: A reliable software defined data plane failure recovery scheme. In *14th International Conference on Network and Service Management, CNSM 2018, Rome, Italy, November 5-9, 2018*, pages 268–274. IEEE Computer Society, 2018.

[13] Israat Haque, Mohammed Nurujjaman, Janelle Harms, and Nael Abu-ghazaleh. SDSense: An agile and flexible SDN-based framework for wireless sensor networks. *The IEEE Transactions on Vehicular Technology*, 68(2):1866 – 1876, February 2019.

[14] Israat Haque and Dipon Saha. SoftIoT: A resource-aware sdn/nfv-based iot network. *The Elsevier Journal of Network and Computer Applications*, 193, Nov 2021.

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

[16] Zaobo He, Yusen Li, Daehee Seo, and Zhipeng Cai. Fedcpd: Addressing label distribution skew in federated learning with class proxy decoupling and proxy regularization. *Information Fusion*, 110:102481, 2024.

[17] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.

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

[19] M. Kulkarni, M. Baddeley, and I. Haque. Embedded vs. external controllers in software-defined iot networks. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, 2021.

[20] Michael P LaValley. Logistic regression. *Circulation*, 117(18):2395–2399, 2008.

[21] Udaya Lekhala and Israat Haque. Piqos: A programmable and intelligent qos framework. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 - May 2, 2019*, pages 234–239. IEEE, 2019.

[22] Guanghao Li, Yue Hu, Miao Zhang, Ji Liu, Quanjun Yin, Yong Peng, and Dejing Dou. Fedhisyn: A hierarchical synchronous federated learning framework for resource and data heterogeneity. In *Proceedings of the 51st International Conference on Parallel Processing*, pages 1–11, 2022.

[23] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[24] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019.

[25] Songtao Lu, Yawen Zhang, and Yunlong Wang. Decentralized federated learning for electronic health records. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5. IEEE, 2020.

[26] 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.

[27] 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.

[28] M. A. Moyeen, Fangye Tang, Dipon Saha, and Israat Haque. SD-FAST: A packet rerouting architecture in SDN. In *15th International Conference on Network and Service Management, CNSM 2019, Halifax, NS, Canada, October 21-25, 2019*, pages 1–7. IEEE, 2019.

[29] Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

[30] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.

[31] Luong Trung Nguyen, Junhan Kim, and Byonghyo Shim. Gradual federated learning with simulated annealing. *IEEE Transactions on Signal Processing*, 69:6299–6313, 2021.

[32] Dipon Saha, Meysam Shojaee, Michael Baddeley, and Israat Haque. An Energy-Aware SDN/NFV architecture for the internet of things. In *IFIP Networking 2020 Conference (IFIP Networking 2020)*, Paris, France, June 2020.

[33] Osama Shahid, Seyedamin Pouriyeh, Reza M Parizi, Quan Z Sheng, Gautam Srivastava, and Liang Zhao. Communication efficiency in federated learning: Achievements and challenges. *arXiv preprint arXiv:2107.10996*, 2021.

[34] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10(1):12598, 2020.

[35] Yandong Shi, Yong Zhou, and Yuanming Shi. Over-the-air decentralized federated learning. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 455–460. IEEE, 2021.

[36] Meysam Shojaee, Miguel C. Neves, and Israat Haque. Safeguard: Congestion and memory-aware failure recovery in SD-WAN. In *16th International Conference on Network and Service Management, CNSM 2020, Izmir, Turkey, November 2-6, 2020*, pages 1–7. IEEE, 2020.

[37] CC Sobin. A survey on architecture, protocols and challenges in iot. *Wireless Personal Communications*, 112(3):1383–1429, 2020.

[38] Fabio A Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. A dataset for breast cancer histopathological image classification. *Ieee transactions on biomedical engineering*, 63(7):1455–1462, 2015.

[39] Fangye Tang and Israat Haque. Remon: A resilient flow monitoring framework. In *Network Traffic Measurement and Analysis Conference, TMA 2019, Paris, France, June 19-21, 2019*, pages 137–144. IEEE, 2019.

[40] Fangye Tang, Meysam Shojaee, and Israat Haque. ACE: an accurate and cost-effective measurement system in SDN, 2022.

[41] Vasiliĭ Borisovich Uvarov. *Mathematical Analysis*. Mir Publishers, 1988.

[42] Zhao Wang, Yifan Hu, Jun Xiao, and Chao Wu. Efficient ring-topology decentralized federated learning with deep generative models for industrial artificial intelligent. *arXiv preprint arXiv:2104.08100*, 2021.

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

[44] Guang Yang, Ke Mu, Chunhe Song, Zhijia Yang, and Tierui Gong. Ringfed: Reducing communication costs in federated learning on non-iid data. *arXiv preprint arXiv:2107.08873*, 2021.

[45] Liangqi Yuan, Lichao Sun, Philip S Yu, and Ziran Wang. Decentralized federated learning: A survey and perspective. *arXiv preprint arXiv:2306.01603*, 2023.

[46] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning, 2021.

[47] 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.

[48] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[49] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *Neurocomputing*, 465:371–390, 2021.