FEDERATED LEARNING IN WIRELESS NETWORKS

by

Xiang Ma

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:

Rose Qingyang Hu, Ph.D.
Major Professor

Todd Moon, Ph.D.
Committee Member

Zhen Zhang, Ph.D.
Committee Member

Bedri Cetiner, Ph.D.
Committee Member

Ziqi Song, Ph.D.
Committee Member

D. Richard Cutler, Ph.D.
Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2024

ABSTRACT

Federated Learning in Wireless Networks

by

Xiang Ma, Doctor of Philosophy

Utah State University, 2024

Major Professor: Rose Qingyang Hu, Ph.D.
Department: Electrical and Computer Engineering

With the advancement of the semiconductor industry, there has been a rapid proliferation of devices on the Internet of Things (IoT), such as smartphones, smart speakers, and intelligent vehicles. Furthermore, the computational capabilities of these IoT devices have also increased significantly. This allows for a significant shift, where more computations occur at the local level rather than in the cloud. Federated learning (FL) is a promising distributed machine learning paradigm that enables clients to execute task computations locally without sharing raw data with a central server. Clients obtain enhanced collective intelligence in FL through model parameter sharing and global aggregation. This approach preserves data privacy among clients and fosters the collaborative construction of a robust model. Despite its promise of addressing privacy concerns, FL faces many challenges, particularly within wireless networks. There are four primary challenges in FL: high communication cost, system heterogeneity, statistical heterogeneity, and privacy and security issues. As the learning process engages many clients while operating within limited wireless channels, this results in considerable communication overhead. The state-of-the-art deep learning algorithm requires the manipulation of millions or even billions of parameters during model training. Transmitting these parameters becomes a bandwidth-intensive process that incurs transmission delays. The phenomenon of system heterogeneity extends its influence on device configurations, deployment scenarios, and connectivity capabilities. Furthermore, statistical heterogeneity becomes prominent through data distribution and model variation divergences. Furthermore, due to the distributed architecture, FL remains vulnerable to attacks inside and outside the system.

This dissertation has explored various algorithms to address the aforementioned challenges. First, the proposed research leverages cutting-edge communication methodologies, including Non-Orthogonal Multiple Access (NOMA), over-the-air computation, and approximate communication techniques. Additionally, gradient compression, client scheduling, and power allocation are used to reduce communication overhead. Incorporating asynchronous FL is instrumental in tackling system heterogeneity by decoupling communication from computation processes. In all scenarios, data distribution takes into account statistical heterogeneity. Ultimately, security and privacy issues are addressed by implementing innovative model update aggregation and individual client initialization schemes.

(136 pages)

PUBLIC ABSTRACT

Federated Learning in Wireless Networks

Xiang Ma

Artificial intelligence (AI) is transitioning from a long development period into reality. Notable instances like AlphaGo, Tesla's self-driving cars, and the recent innovation of ChatGPT stand as widely recognized exemplars of AI applications. These examples collectively enhance the quality of human life. An increasing number of AI applications are expected to integrate seamlessly into our daily lives, further enriching our experiences.

Although AI has demonstrated remarkable performance, it is accompanied by numerous challenges. At the forefront of AI's advancement lies machine learning (ML), a cutting-edge technique that acquires knowledge by emulating the human brain's cognitive processes. Like humans, ML requires a substantial amount of data to build its knowledge repository. Computational capabilities have surged in alignment with Moore's law, leading to the realization of cloud computing services like Amazon AWS. Presently, we find ourselves in the era of the IoT, characterized by the ubiquitous presence of smartphones, smart speakers, and intelligent vehicles. This landscape facilitates decentralizing data processing tasks, shifting them from the cloud to local devices. At the same time, a growing emphasis on privacy protection has emerged, as individuals are increasingly concerned with sharing personal data with corporate giants such as Google and Meta. Federated learning (FL) is a new distributed machine learning paradigm. It fosters a scenario where clients collaborate by sharing learned models rather than raw data, thus safeguarding client data privacy while providing a collaborative and resilient model.

FL has promised to address privacy concerns. However, it still faces many challenges, particularly within wireless networks. Within the FL landscape, four main challenges stand out: high communication costs, system heterogeneity, statistical heterogeneity, and privacy and security. When many clients participate in the learning process, and the wireless communication resources remain constrained, accommodating all participating clients becomes very complex. The contemporary realm of deep learning relies on models encompassing millions and, in some cases, billions of parameters, exacerbating communication overhead when transmitting these parameters. The heterogeneity of the system manifests itself across device disparities, deployment scenarios, and connectivity capabilities. Simultaneously, statistical heterogeneity encompasses variations in data distribution and model composition. Furthermore, the distributed architecture makes FL susceptible to attacks inside and outside the system.

This dissertation presents a suite of algorithms designed to address the challenges effectively. Mew communication schemes are introduced, including Non-Orthogonal Multiple Access (NOMA), over-the-air computation, and approximate communication. These techniques are coupled with gradient compression, client scheduling, and power allocation, each significantly mitigating communication overhead. Implementing asynchronous FL is a suitable remedy to solve the intricate issue of system heterogeneity. Independent and identically distributed (IID) and non-IID data in statistical heterogeneity are considered in all scenarios. Finally, the aggregation of model updates and individual client model initialization collaboratively address security and privacy issues.

To all the people whom I love and who love me...

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude and appreciation to my advisor, Prof. Rose Qingyang Hu. She has provided tremendous support and guidance throughout my PhD studies. She has taught me how to conduct excellent research, teach in the classroom, and give research talks to the audience through her words and deeds. The mentorship I have received from her will benefit me for the rest of my life.

I also want to thank Prof. Todd Moon, Prof. Zhen Zhang, Prof. Bedri Cetiner, and Prof. Ziqi Song for serving as my PhD committee members and for providing valuable comments and advice during my PhD study. Their help has greatly aided me throughout my journey in pursuing a PhD.

I am also grateful to Prof. Haijian Sun and Prof. Yi Qian, who have helped me with many research projects. They discussed with me on my research paper writing and career selection, which made me feel supported.

For my friends, I want to thank Dr. Qun Wang, Dr. Fuhui Zhou, Dr. Haishan Yang, Dr. Maowei Hu, Dr. Haofan Cai, Dr. Tianyi He, Dr. Nathan Carruth, Dr. Yiming Zhao, Dr. Yanjun Wan, Dr. Tao Zhang, Dr. Al Forsyth, Xiankun Yan, Qianqiu Longyang, Zhao Ming, Jie Ren, Jiahao Wen, Zichao Wang, Xingyi Zhao, Yili Zhang, Jinghui Jiang, Yuan Zhou, Qianshun Wei, Liming Tan, Keyang Wu, Xudong Zhang, Wei Chuang, Corey Zhao, Billy Chen, Tony Hsiao, Sam Christensen, Nathan Hult, Jack Greene, Manijeh Nouraei, Kirk Petersen, Casey Bond, Emron Bardsley, and Jace Bingham. They helped me with research discussions or life directions. With their help, I have navigated the challenges of Covid-19 and the experience of being away from home for five years.

I also want to thank my mother Dongju Leng, my elder sister Lanlan Ma, my brother-in-law Wencheng Wang, my niece Fuyi Wang, and my nephew Fushuo Wang. Thanks for your understanding and encouragement to make my dream come true. I also know that my father, Xingwei Ma, in heaven, has always blessed my life.

Lastly, I want to thank Mengying Jiang. We met during our PhD studies at USU, where we got to know each other. She has provided me with great help and support during my studies, research, job search, and graduation. Without her support, I cannot imagine what would have happened to me. Knowing her is a great blessing.

This work has been supported by the National Science Foundation under grants CNS-2007995 and ECCS-2139508.

Xiang Ma

CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

NOTATION

FL system

D	dataset
K	the number of clients scheduled in each round
M	the total number of clients
T	FL training rounds
g	FL model gradient information
w	FL model weight information
η	learning rate

Channel Model

B	bandwidth
R	data rate
d	distance between the server and client
h	fading factor
n	noise
p	transmission power
α	path-loss exponent
λ	wavelength
σ^2	noise variance

ACRONYMS

AFL	Asynchronous Federated Learning
AI	Artificial Intelligence
AirComp	Over-the-air Computation
BER	Bit Error Rate
CNN	Convolutional Neural Networks
ECC	Error Correction Code
FEC	Forward Error Correction
FedAvg	Federated Averaging Algorithm
FL	Federated Learning
IID	Independent and Identically Distributed
IoT	Internet of things
MAC	Multiple Access Channel
ML	Machine Learning
MSE	Mean Square Error
NOMA	Non-Orthogonal Multiple Access
PS	Parameter Server
QAM	Quadrature Amplitude Modulation
SGD	Stochastic Gradient Descent
SNR	Signal-to-noise Ratio
TDMA	Time Division Multiple Access
UAV	Unmanned Aerial Vehicle

CHAPTER 1

INTRODUCTION

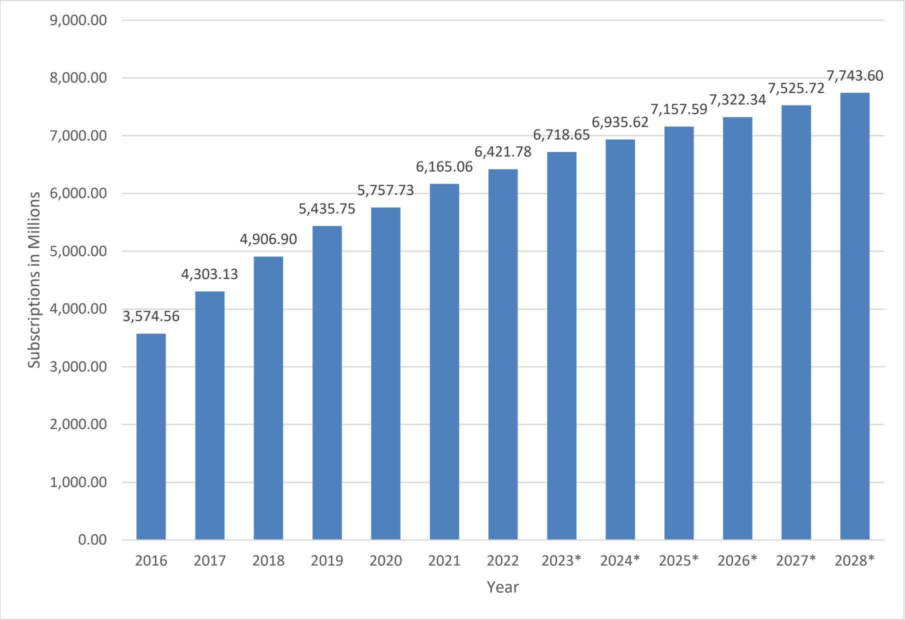
Federated learning (FL) was first proposed by Google in 2016 to train a collaborative model without sharing data from individual mobile devices [1]. It has been implemented on the Google keyboard, which is called Gboard. FL is a new machine learning (ML) architecture that allows each mobile device to keep data locally and train the model on each device without sharing data with other devices. To gain collective intelligence, the learned model is shared instead. In this way, the data privacy of mobile devices is protected while a global model can still be learned.

1.1 Background

With the development of the semiconductor industry, the computational capacities of edge mobile devices, such as smartphones, smartwatches, and unmanned aerial vehicles (UAVs), have been increasing rapidly. The number of mobile devices is also growing tremendously. According to Ericsson [2], the number of smartphone subscriptions worldwide will double from 2016 to 2025, as shown in Fig. 1.1.

The large computation capacities of edge mobile devices allow the computation to occur on the edge mobile device rather than on the cloud or central server. This allows for the training of ML tasks locally. However, ML needs a substantial amount of data to train the model in order to achieve acceptable performance. The AlphaGo is trained using a KGS data set with 29.4 million moves from 160,000 games [3]. GPT-3 [4] used 570 GB text data after filtering from a 45 TB data set. These data come from public resources or are collected from users. During the data collection and transmission process, users' privacy may be compromised.

The public has been increasingly concerned about their privacy. The General Data Protection Regulation (GDRP) [5] was approved by the European Union (EU) in 2016 and



[2]

Fig. 1.1: Number of smartphone mobile network subscriptions worldwide

went into effect in 2018. And most recently, Meta (previous Facebook) agreed to pay $725 million to settle a privacy lawsuit. More actions are needed to protect the privacy of users in the age of artificial intelligence (AI). A new ML diagram, termed Federated Learning (FL), emerges to protect users' privacy while training a robust model. This is achieved by sharing the trained model rather than raw data among users.

1.2 FL Applications

Due to the privacy protection features in distributed learning, FL has been applied to multiple fields. In [6], the authors reviewed the application of FL in mobile devices, industrial engineering, and healthcare. The prediction of keyboard action is performed in [7], where out-of-vocabulary words are learned and predicted using the FL method. Additionally, human trajectory prediction can be done without sharing the user's real trajectory with the server in [8]. In industrial engineering, monitoring data can be learned and trained with FL to build a robust global ML model [9]. UAV network jamming attacks can be detected using FL without sharing UAV trajectory data [10]. FL uses patient data available in each medical institute in healthcare and breaks down barriers through different hospitals [11]. In [12], the researchers applied FL to predict the mortality rate of patients with heart disease using data from electronic health records.

1.3 System Model

FL system usually consists of a central parameter server (PS) and a large number of clients. The learning process runs iteratively between the server and the clients, typically spanning multiple rounds. The model learned in the previous round serves as the starting point for the learning in the subsequent round. Each round has four primary steps, as shown in Fig. 1.2. First, in step (S1), the server transmits the current global model to the clients. Second, in (S2), clients utilize the global model as their starting point and apply optimization methods such as stochastic gradient descent (SGD) to derive updated local models. Then, in step (S3), the clients upload their updated local models to the server. Finally, in step (S4), the server aggregates the local models received to create a new global model, which serves as the basis for the next round. This iterative process continues until the model converges. Here, the FL system consists of one PS and M clients. Each client m has a dataset D_m.

FL aims to train a global model with data distributed on each client. The objective function can be defined as

$$\min_{\boldsymbol{w}\in R^d} f(\boldsymbol{w}), \tag{1.1}$$

where $\boldsymbol{w}$ is the model parameter, d is the model size, $f(\boldsymbol{w}) = \frac{1}{|D|}\sum_{i=1}^{|D|} f_i(\boldsymbol{w})$, $|D|$ is the size of the dataset D, which is the collection of all dataset D_m. $f_i(\boldsymbol{w}) = \ell(\boldsymbol{x_i}, y_i; \boldsymbol{w})$ is the loss function used to capture the error between the data sample $(\boldsymbol{x_i}, y_i)$ and the mapping made by $\boldsymbol{w}$. Since the data are distributed among M clients, the objective Eq. (1.1) can be rewritten as

$$f(\boldsymbol{w}) = \sum_{m=1}^{M} \frac{|D_m|}{|D|} F_m(\boldsymbol{w}), \tag{1.2}$$

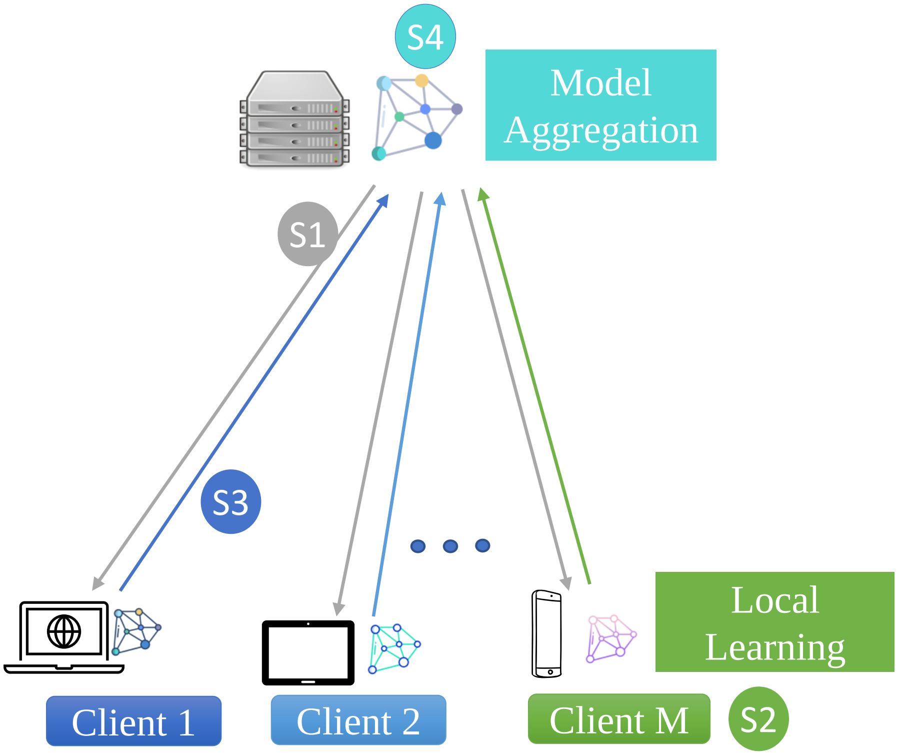


Fig. 1.2: System model

where, $F_m(\boldsymbol{w}) = \frac{1}{|D_m|}\sum_{i\in D_m} f_i(\boldsymbol{w})$. The centralized objective Eq. (1.1) is transformed in a distributed manner in Eq. (1.2).

In each FL round, the client performs local training based on the local data and the most recently received global model, that is

$$w_t^m = w_t - \eta \nabla F_m(w_t), \tag{1.3}$$

where at round t, $\nabla F_m(w_t)$ is the gradient of $F_m(w_t)$, η is the learning rate, w_t is the received global model, w_t^m is the local model on client m.

The server aggregates the uploaded local model to build a global model as

$$w_{t+1} = \sum_{m=1}^{M} \frac{|D_m|}{|D|} w_t^m. \tag{1.4}$$

Here, $\frac{|D_m|}{|D|}$ is the contribution of the client m to the global model, and $\sum_{m=1}^{M}\frac{|D_m|}{|D|} = 1$. This is termed the Federated Raw (FedRaw) method here.

Another method of aggregating the global model is federated SGD (FedSGD), where the local gradient rather than the local model is uploaded to the central server. The local gradient is calculated as

$$g_t^m = \nabla F_m(w_t). \tag{1.5}$$

Subsequently, the global model is computed using the most recent gradient and the global model from the previous round, i.e.,

$$w_{t+1} = w_t - \eta \sum_{m=1}^{M} \frac{|D_m|}{|D|} g_t^m. \tag{1.6}$$

The main difference between FedSGD and FedRaw is the parameters uploaded by the clients. In FedSGD, the local gradient information is uploaded, and the server performs gradient aggregation and updates the global model as in Eq. (1.6). Clients upload the local weight information to FedRaw.

This method is under ideal conditions where all clients participate in the learning.

When the channel resource is constrained, and only a fraction of clients are allowed to participate in the learning, the classical algorithm to implement FL is Federated Averaging (FedAvg). The pseudo-code is given in Algorithm 1.

Algorithm 1 FederatedAveraging

```
**Server executes:**
  initialize $w_0$
  **for** each round t=1,2,... **do**
    $K \leftarrow \max(C \cdot M, 1)$
    $S_t \leftarrow$ (random set of $K$ clients)
    **for** each client $m \in S_t$ **in parallel do**
      $w_{t+1}^m \leftarrow$ ClientUpdate($m$, $w_t$)
    **end for**
    $|D_{S_t}| \leftarrow \sum_{m \in S_t} |D_m|$
    $w_{t+1} \leftarrow \sum_{m \in S_t} \frac{|D_m|}{|D_{S_t}|} w_{t+1}^m$
  **end for**
**ClientUpdate**($m$, $w$): // *Run on client m*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_m$ into batches of size $B$)
  **for** each local epoch $i$ from 0 to $E$ **do**
    **for** batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
    **end for**
  **end for**
  return $w$ to server
```

The FedAvg algorithm addresses the limited channel problem when there are many clients in the FL system. When $C = 1$, FedAvg becomes FedRaw, which means that all clients participate in the learning process. Otherwise, only a fraction of clients can upload their locally learned models. When $C = 1$, and $E = 1$, FedAvg becomes FedSGD.

In summary, the weights or gradients of the participating clients can be uploaded to the server. After the gradients are uploaded, the server aggregates the received parameters and calculates the most recent global model based on Equation (1.6). The server always sends the most recent global model to the clients for the next round of training.

1.4 Existing Problems

While federated learning is a promising technique to address privacy issues in distributed machine learning, there are still many challenges. In [13], the authors mentioned four key challenges associated with FL: high communication cost, system heterogeneity, statistical heterogeneity, and privacy concerns.

High communication costs may appear from several perspectives. First, the FL system potentially has many clients, e.g., millions of smartphones, and communication becomes the system's bottleneck. Second, state-of-the-art neural networks require millions, even billions, of model parameters to train the model. This results in a high communication overhead when transmitting the model parameters. The heterogeneity of the system is reflected in the variety of clients and their connectivity capabilities. This could be caused by differences in hardware, network infrastructure, power, storage, computational capacity, and communication capabilities. System heterogeneity makes it difficult for the server to manage the learning process. For statistical heterogeneity, the data across clients may be non-independent and identically distributed (non-IID). One of the primary distinctions between Federated Learning (FL) and classical distributed learning lies in the data source. In FL, data are typically collected from local environments instead of being assigned by the server. The distributed nature of the Federated Learning (FL) system can cause privacy and security concerns. There is a risk that certain clients could be compromised and exploited to launch attacks on the entire system. To protect learning performance, it is essential to implement additional security measures. In addition, gradient information has been shown to potentially reveal some aspects of privacy. Malicious actors like eavesdroppers or dishonest parameter servers might exploit gradient information to infer sensitive raw data.

1.5 Literature Review

Numerous efforts have been made to tackle the challenges of Federated Learning (FL), as outlined above. This section focuses on a comprehensive examination of existing research and efforts.

1.5.1 Communication Cost

The massive number of clients who participate in the learning process and ML models to train the data on the clients result in high communication costs. To mitigate communication delays in Federated Learning (FL) within wireless networks, various approaches have been explored, including 1) decreasing communication frequency, 2) implementing compression techniques, 3) utilizing advanced communication methods, and 4) employing resource scheduling strategies. These schemes aim to enhance the efficiency of FL in wireless network settings.

1.5.1.1 Reducing communication frequency

In FL, clients tend to communicate with the server more frequently to gain collective intelligence from other clients. In [1], the authors proposed a communication-efficient algorithm FederatedAveraging (FedAvg). In FedAvg, only a fraction of clients are selected to participate in the learning process in each round, and a small mini-batch of data is used in each round. Furthermore, multiple local iterations are performed in each FL round to reduce the communication frequency. The hierarchical FL of the client edge cloud is presented in [14]. By introducing the edge layer, clients can send the local model to the edge server for aggregation rather than directly sending it to the cloud server. Therefore, communication with the cloud server is reduced. The device-to-device (D2D) training of FL is also applied in [15]. No server is involved in this design, and clients communicate directly with neighboring clients. This expands the coverage areas of the client distribution and increases channel frequency reuse.

1.5.1.2 Compression schemes

Model compression can be applied to reduce the transmission of the number of model parameters. In [16], the authors proved that 99% of gradients can be dropped in distributed gradient descent with little or no performance loss. Gradient quantization [17] can be used after gradient decrease. The model updates are forced to be sparse and low-rank in [18].

Model updates can be learned from a restricted space. In this way, only a small fraction of gradients need to be transmitted to the server.

1.5.1.3 Advanced communication schemes

In wireless networks, advanced communication can be employed to improve communication efficiency. Non-orthogonal multiple access (NOMA) is used in FL to allow multiple users to transmit at the same channel simultaneously [19]. Over-the-air computation (AirComp) can also be applied in FL to aggregate the model faster in [20]. The superposition property of multiple-access wireless channels allows for multiple signal aggregation during transmission. Therefore, multiple users are allowed to transmit their local models simultaneously. Additionally, FL only needs a global model on the server side, and AirComp saves time by decoding individual models.

1.5.1.4 Client Scheduling

Clients can be scheduled to participate in the learning process according to channel conditions or the importance of achieving optimal performance in wireless networks. Client scheduling policies are explored in [21] when only a subset of clients can be scheduled to participate in the learning process. The authors considered three practical approaches: random scheduling, round robin, and proportional fairness based on channel conditions. Different schemes require different learning rounds to achieve convergence under a high or low signal-to-interference-plus-noise ratio (SINR) threshold. In [22], a joint client scheduling and resource allocation algorithm is proposed. The bandwidth is allocated to the scheduled clients to achieve the best performance.

1.5.2 System Heterogeneity

As mentioned above, system heterogeneity describes the varieties of client computation, storage, and connectivity in the FL system. To handle the system heterogeneity problem, 1) active client sampling and 2) asynchronous communication can be employed.

1.5.2.1 Active client sampling

FL aggregation can be triggered periodically without waiting for all clients to complete their local learning. The straggler problem can be solved with some existing methods. However, the straggler is less likely to have the opportunity to access the channel. In [23], the researchers proposed a straggler-resistant algorithm by gradually increasing the number of participants. Channel condition-based client scheduling is proposed in [24]. A subset of clients with the highest channel gain are selected to participate in the learning in each round.

1.5.2.2 Asynchronous communication

The classical FL works in synchronous mode. The computation in the subsequent round runs after completing all communications. Asynchronous communication is applied in [25] [26] to decouple computation and communication. The server aggregates the global model once it receives a local model from clients rather than waiting for all clients.

1.5.3 Statistical Heterogeneity

Since client data are collected from the local environment, the data distribution can be very divergent. In addition, some clients may want a personalized rather than an identical global model. The weight divergence in [27] explains FL with non-IID data. The authors created a small subset of data shared globally between all clients to improve training performance. Also, new algorithms should work on both IID and non-IID data settings to verify their effectiveness.

Personalized FL is applied in [28] to provide personalization of clients in FL. The authors presented an adaptive, personalized FL algorithm to train local models on clients while contributing to the global model. The personalized model for each client is the mixing of the global model and the local model with some mixing weight. Better performance can be achieved when the mixing weight is tuned adaptively.

1.5.4 Security and Privacy

FL operates as a distributed system that is vulnerable to internal threats. External attackers can intrude and control some clients, mislead the global model, and leak client privacy to dishonest servers or eavesdroppers.

In [29], verifiable outlier detection, secure distance computation, and secure model aggregation are used to make the Byzantine-resilient FL system. To mitigate backdoor attacks, a model pruning method is proposed in [30] to remove redundant neurons and adjust the extreme weight value method of the model. The average attack success rate can be significantly reduced with little loss of test accuracy.

Differential privacy (DP) has become an almost standard method for protecting client privacy in distributed systems. In [31], DP is used to protect the privacy of patient health data in FL. DP is used in wireless FL in [32].

1.6 Dissertation Outline

This dissertation focuses on the problems of federated learning in wireless networks, including high communication costs, system heterogeneity, statistical heterogeneity, and security and privacy issues. Chapters 2 to 5 focus on methods to reduce communication costs. Chapter 6 presents a novel asynchronous federated learning architecture to solve system heterogeneity problems. Chapter 7 proposes model update-based aggregation and individual client model initialization schemes to enhance FL's security and privacy. The statistical heterogeneity problems are considered in all conditions by applying the proposed algorithm to non-IID data.

In Chapter 2, NOMA reduces communication time by allowing multiple users to upload their local model parameters in FL. To our knowledge, we are the first to apply NOMA in FL. The power allocation in NOMA makes the capacity vary for different users. Therefore, adaptive gradient compression is used to satisfy the channel capacity constraints of the transmitted message.

Chapter 3 extends the FL with NOMA by considering user scheduling and power allocation to improve learning performance. A joint user scheduling and power allocation scheme is presented to achieve the maximum weight sum rate of the system. This allows users to transmit more messages under compression. Better learning performance is achieved with optimal joint user scheduling and power allocation.

Chapter 4 introduces the approximate communication for the transmission of the FL model. The ML system is found to be error-resilient based on multiple pieces of evidence. The transmission of the FL model parameter, especially through wireless links, can cause tremendous errors. Motivated by the gradient clipping for gradient exploding problems, we constrain the gradient value in a small range by employing the gradient distribution knowledge. Transmission error is also limited to a small range under the FL system error resilience capabilities. Then, the FL model is transmitted approximately without forward error correction and packet retransmission.

In Chapter 5, the over-the-air computation (AirComp) is applied to reduce the communication and computation time in FL. Like NOMA, AirComp allows multiple users to transmit simultaneously on the same channel. Unlike NOMA, the aggregation of the messages takes place in the air and cannot be decoded on the receiver side. This fits well for FL, which only needs the aggregated results. Several different user scheduling schemes have also been explored in FL with AirComp.

Chapter 6 focuses on security and privacy issues in FL. FL is a distributed system that is vulnerable to Byzantine attacks. A model update-based (MUB) aggregation makes the aggregated information less susceptible to Byzantine attacks. The individual client model initialization (ICMI) scheme helps hide each client's initial model. Combined with MUB, ICMI can hide the local model during learning, protecting private data from membership-inference attacks.

In Chapter 7, we consider the system heterogeneity problem. The computational capabilities vary between clients, which may have straggler issues. Due to FL's distributed training nature, this will affect the model convergence training time. Asynchronous FL

(AFL) is applied to speed up the learning process. The client scheduling and novel model aggregation algorithms make model training faster. Model staleness problems in AFL are addressed.

Chapter 8 concludes this dissertation and discusses future research directions. It will explore the ongoing research on FL in wireless networks.

CHAPTER 2

Adaptive Federated Learning with Gradient Compression in Uplink NOMA

2.1 Introduction

This chapter considers the transmission of the FL model in wireless networks using NOMA. NOMA improves spectrum efficiency and reduces transmission time. Because NOMA allocates power for successive interference cancellation decoding at the receiver, gradient compression is performed to satisfy channel capacities.

FL is a distributed ML technique. The learning process is carried out round by round. The server first transmits the global model to the clients in each round. Then, the clients will send the learned local model back to the server after local training. The model parameter information is shared between the server and clients. In wireless networks, time division multiple access (TDMA) [33] or frequency division multiple access (FDMA) [34] allows multiple users to access the time/frequency. Under TDMA, users are assigned different time slots for the whole channel. Each user needs to wait for other users to finish their transmission until it is scheduled. Correspondingly, for FDMA, the entire channel band is divided into small pieces for each user. The user can use the scheduled channel band all the time. Unlike TDMA or FDMA, NOMA [35] allows multiple users to transmit their signals in the same channel simultaneously. The channel and time resources are shared. To successfully decode the message received from each user, the power allocation on the transmitter side and successive interference cancellation (SIC) [36] at the receiver side are applied. Different transmission power and channel conditions cause the channel capacity to differ. Once the learning model is determined, the transmission information size is specified. To satisfy the channel capacities, gradient compression is needed to transmit the gradients in a fixed time duration.

In [37], deep gradient compression is applied to reduce the model size for distributed learning. It was found that 99% of the gradient is redundant. 1-bit gradient quantization [38] is used to explore the low-bit expression of gradients in speech DNNs. These works did not consider the constraints of practical communication networks. In [39], a multi-access channel (MAC) transmits messages simultaneously, analog or digital. The model is aggregated over the air to get the model average. However, the researchers fail to consider the effect of wireless fading channels, making channel conditions impractical. In addition, the projection of gradients transmitted over MAC could result in a higher bit error rate (BER). Here, we consider a capacity-limited fading channel in the uplink of the FL system. The gradient size is adaptively compressed based on the channel capacity. Gradient sparsification is applied to remove redundant information. Gradient quantization can be used to represent gradients with fewer bits.

The contributions of this chapter are summarized as follows:

- This is the first time NOMA has been applied for FL model transmission in wireless networks. NOMA is a favorable selection compared to TDMA in the time perspective.
- Adaptive gradient compression in the NOMA uplink helps to satisfy channel capacity constraints and reduce communication time.
- The effectiveness of the proposed scheme is verified on several datasets. The results show that the communication time for FL is reduced by at least 7× without or with a slight loss in test accuracy.

2.2 System Model

2.2.1 FL Model

Here, we consider an FL model similar to that described in Chapter 1.2. The uploaded information here is the local gradients, and only a fraction of clients are allowed to upload in each round due to channel constraints. Other than that, we use the same notation as in Chapter 1.2.

2.2.2 Uplink NOMA Transmission

In wireless transmission, the link from the client to the server is usually called an uplink, and correspondingly, the link from the server to the client is a downlink. The uplink in the FL system is time-consuming when a large number of clients are connected to the server. However, the downlink can be broadcast to all clients from the server, making it time-efficient. Here, we consider NOMA transmission on the uplink and broadcast on the downlink.

Here, a fading channel is assumed. The channel is assumed to remain constant within the round time but varies across rounds. To simplify the analysis, the time t is omitted here. The channel gain between client k and server is $h_k = L_k h_0$, where L_k is the large-scale fading and h_0 is the small-scale fading. L_k is assumed to follow the free space path loss model $L_k = \frac{\sqrt{\delta_k}\lambda}{4\pi d_k^{\alpha/2}}$, where δ_k is the gain of the transmitter and receiver antenna, λ is the wavelength, d_k is the distance between client k and server, and α is the path loss exponent. h_0 is assumed to follow a normal Gaussian distribution $h_0 \sim \mathcal{N}(0, 1)$.

The transmitted gradient information g_k^t is normalized as s_k^t, where $||s_k^t|| = 1$. According to the NOMA principle, the selected fraction of clients share the same bandwidth simultaneously. The signals transmitted from different users are superposed during air transmission [40]. Here, we consider the K clients selected in each round. The received signal at the server side can be written as:

$$y = \sum_{k=1}^{K} \sqrt{p_k} h_k s_k + n, \tag{2.1}$$

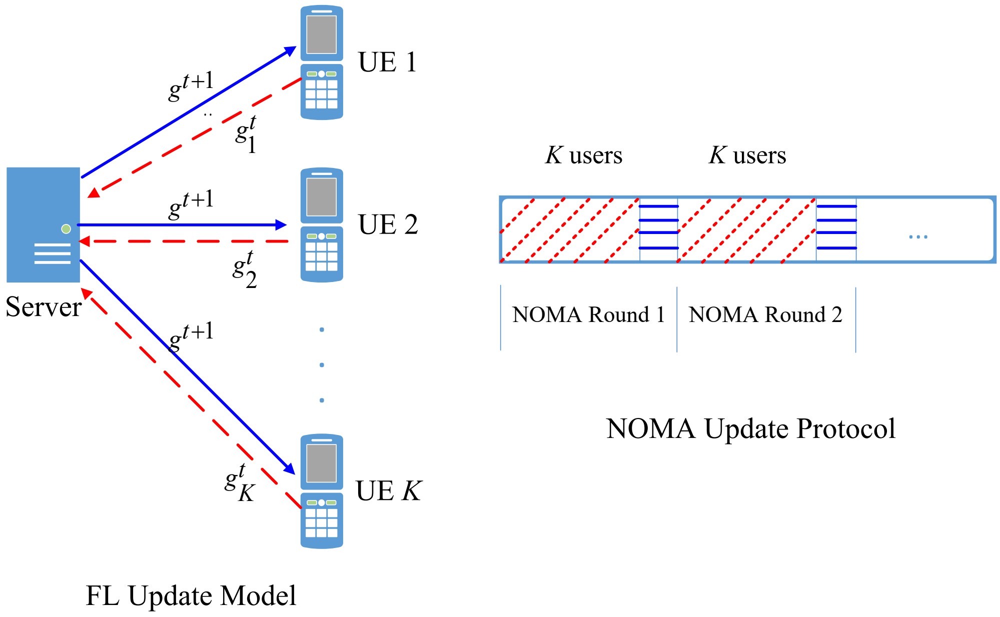
where p_k is the transmission power of client k, and $n \sim \mathcal{N}(0, \sigma^2)$ is the additive noise.

SIC is executed on the receiver side to successfully decode individual s_k from the received signal y. The process of executing SIC is described below. The server distinguishes different users using the power strength of the received signal. The server will decode the client signal with the strongest power strength by considering other signals as interference. Then, the server subtracts the decoded signal from the superposed signal and decodes the next strongest signal. This process continues until the server decodes all signals. Without loss of generality, we assume the power of the received signal $p_1 h_1^2 > p_2 h_2^2 > \cdots > p_K h_K^2$. The achievable data rate for user k is

$$R_k = \log_2 \left\{ 1 + \frac{p_k h_k^2}{\tau(\sum_{j=k+1}^{K} p_j h_j^2 + \sigma^2)} \right\}, \forall k = \{1, \dots K-1\}, \tag{2.2}$$

where $\tau > 1$ stands for performance degradation due to imperfect channel estimation and decoding errors. The data rate for the last decoded user K is $R_K = \log_2(1 + \frac{p_K h_K^2}{\tau \sigma^2})$.

For each client participating in the parameter uploading process, the maximum number of transmission bits for the user k is $m_k = B R_k t_k$, where B is the uplink transmission bandwidth and t_k is the transmission duration. Fig. 2.1 illustrates the FL system and the NOMA uplink NOMA. The red lines indicate the uplink, where K clients are scheduled to transmit their local parameters using NOMA. The blue lines indicate the downlink to the broadcast.



NOMA Update Protocol

FL Update Model

Fig. 2.1: FL update and NOMA update model

2.3 Adaptive Model Compression with NOMA Transmission

The section considers user scheduling to select K clients to participate in the NOMA uplink process first. Once the K users are determined, adaptive gradient compression is performed on each client based on the channel capacities before transmission.

2.3.1 NOMA Scheduling

The selection criteria for the K clients consider 1) NOMA fairness and 2) time budget.

1) NOMA fairness: the server performs a weighted average to the received model parameters. An effective update capacity is defined to account for the actual contribution for the weighted average,

$$R_k^{ef} = \frac{BR_k t_k}{|D_k|}. \tag{2.3}$$

When the data rates of clients vary, FL using NOMA experiences performance degradation. To make sure each user has a fair update, Jain's fairness index is applied, which is defined as:

$$J_u = \frac{(\frac{1}{K}\sum_{k=1}^{K} R_k^{ef})^2}{\frac{1}{K}\sum_{k=1}^{K}(R_k^{ef})^2}. \tag{2.4}$$

When fairness is maximized, J_u should be close to 1. A scheduling algorithm as in [41] is applied to ensure that J_u gets close to 1.

2) Time budget: NOMA is a synchronous system, and FL waits for the K clients to finish uploading. This means that the server needs to wait for the selected K clients to finish their local computation and then start the transmission simultaneously. Since computational capabilities vary between clients, the server will first select the clients who completed the local computation.

2.3.2 Adaptive Model Compression

The ML model parameter is fixed when the architecture of the model is determined. When the gradient size exceeds the maximum allowable transmission size m_k, gradient compression can be used. Gradient compression is executed adaptively to satisfy the channel capacity of different clients and to transmit the parameter as much as possible. Assuming the total number of bits of the model is G, the compression rate for the user k is $r_k = \max\{\frac{G}{m_k}, 1\}$. When $r_k = 1$, it means $G < m_k$, and all gradient parameters are transmitted.

2.3.2.1 Gradient Quantization

A 32-bit floating-point number typically represents each model parameter in ML. Transmitting full-length parameters tends to be expensive. Existing work [37] has shown that good convergence of learning models with slight performance loss can be achieved with gradient quantization. The ML system is resilient to the "rounding errors" caused by quantization.

In empirical studies [42], the gradients are found to be distributed within the $(-1, 1)$ range. The DoReFa scheme [43] is suitable for compressing gradients. We use $r_k^q = \max\{\frac{G}{m_k}, 1\}$ to represent the compression rate of quantization. Then, the number of bits used to represent a gradient value is $b_k^q = \lfloor \frac{1}{r_k^q} 32 \rfloor$, where $\lfloor \cdot \rfloor$ takes the floor operation. In DoReFa, the mapping from a 32-bit floating point number to a quantized number is

$$q_k(x) = \frac{1}{a} \lfloor ax \rceil, \tag{2.5}$$

where $\lfloor \cdot \rceil$ rounds to the nearest integer, x is the gradient value, and $a = 2^{b_k^q} - 1$. And $q_k(x)$ is the quantized gradient value, represented by b_k^q bits.

Since each gradient parameter is quantized with a compression rate r_k^q, the transmission size of all parameters is also quantized with a rate r_k^q.

2.3.2.2 Gradient Sparsification

Sparsification refers to the approach of sparsifying the gradient vector and setting a fraction of the gradient values as zero. Only the remaining gradients will be sent. Empirical experiments [16] have shown that a large proportion of gradient updates in a distributed SGD are redundant. We keep the important gradients and leave the less critical gradients as zero. The magnitude of the gradient is one of the metrics used to measure its importance. The general rule is to use a threshold to perform the sparsification. Gradients with a magnitude more significant than the threshold will be kept. Otherwise, they will be mapped to zero. After sparsification, the nonzero gradients with their index information will be uploaded to the server. The server will reconstruct the gradient vector based on the information received and perform the aggregation.

Since the indices of the nonzero gradient values are also transmitted, the sparsification rate can be calculated as

$$\frac{G}{r_k^s} + \frac{G}{r_k^s 32}\bar{b}_k = m_k, \tag{2.6}$$

where $r_k^s = \max\{r_k^s, 1\}$ is the sparsification rate, and $\bar{b}_k$ is the number of bits used for encoding the index information. $\frac{G}{r_k^s 32}$ denotes the number of gradients transmitted when the 32-bit floating point number represents each gradient.

Rather than directly transmitting the index information of the nonzero gradient, the index's relative distances between adjacent nonzero gradients are transmitted. Additionally, we use a nonlinear coding method called the Golomb code [44] to encode the relative distance, which can further save space with variant-length bit representation. The average number of bits used to encode the relative distance is

$$\bar{b}_k = b_k^* + \frac{1}{1 - (1 - r_k^s)^{2^{b_k^*}}}, \tag{2.7}$$

where, $b_k^* = 1 + \lfloor \log 2(\frac{\log(\phi - 1)}{\log(1 - r_k^s)}) \rfloor$, $\phi = \frac{1+\sqrt{5}}{2}$. r_k^s and $\bar{b}_k$ are coupled in Eq. (2.6) and Eq. (2.7), they can be solved together. When r_k^s is obtained, the sparsification threshold can be calculated.

Compared with the Algorithm in Chapter 1.2, the proposed algorithm applied NOMA in practical wireless networks. Gradient compression is used to satisfy the channel capacity.

2.4 Simulation

2.4.1 Simulation Settings

This section describes our communication model settings and the FL learning hyperparameter settings. We then present the simulation results on three datasets: MNIST, FEMNIST, and Sent140.

The proposed NOMA-based transmission with the gradient compression scheme will be evaluated from both a round and communication-time perspective. The TDMA-based original FedAvg will serve as the baseline.

First, the channel parameters are given as follows. The bandwidth for uplink is $B = 5$ MHz, the path loss exponent $\alpha = 3$, and the additive noise power density $\sigma^2 = -174$ dBm/Hz. The number of users allowed to transmit via the NOMA link in each round is assumed to be $K = 10$ or $K = 20$. All users are randomly distributed in a disk region with a radius equal to 500 meters. The fading channel gain h_k is calculated based on these settings. The transmission power is assumed to be $p_k = 0.1$ watts for all users. And the duration of the uplink transmission $t_k = 0.5$ s. The aggregated parameter is broadcast to all clients without compression for the downlink. The transmission time is calculated as $T_d = \max_k \frac{G}{B_d \log 2(1+p_d \gamma_k)}$, where $B_d = 10$ MHz is the downlink bandwidth, $p_d = 2$ watts is the downlink power, and γ_k is the signal-to-noise ratio (SNR) from the server to the client k.

Three different datasets are explored to make the proposed scheme more convincing, including two image datasets, MNIST (Modified National Institute of Standards and Technology) [45] and Federated Extended MNIST (FEMNIST) [46], and one text dataset, Sentiment140 (Sent140). Image classification tasks are performed on the two image datasets with the same LeNet-300-100 model. On the contrary, the text sentiment analysis task is performed on Sent140 with a long- and short-term memory (LSTM) classifier [47]. The system heterogeneity is reflected in the non-IID data settings, which is the sample and the number of samples varying across the clients. Here, a similar data division algorithm from [47] is employed. The statistics of the datasets are summarized in Table 2.1.

For FL hyperparameter settings, the batch size $\mathcal{B} = 10$ is used in all datasets. The

Table 2.1: Statistics of Datasets

Dataset	No. of Parameters (P)	No. of Devices (N)	No. of Data (D)
MNIST	266,610	1,000	69,035
FEMNIST	266,610	200	18,345
Sent140	243,861	660	40,783

learning rate and communication rounds are fixed for each dataset but vary for different datasets to optimize learning performance. Specifically, a learning rate $\eta = 0.001$ and the communication round $T = 100$ are used for MNIST, $\eta = 0.003$ and $T = 300$ for FEMNIST, and $\eta = 0.05$ and $T = 100$ for Sent140. The testing is performed on the test dataset. The test accuracy is used as the metric to measure the learning performance in different scenarios.

2.4.2 Simulation Results

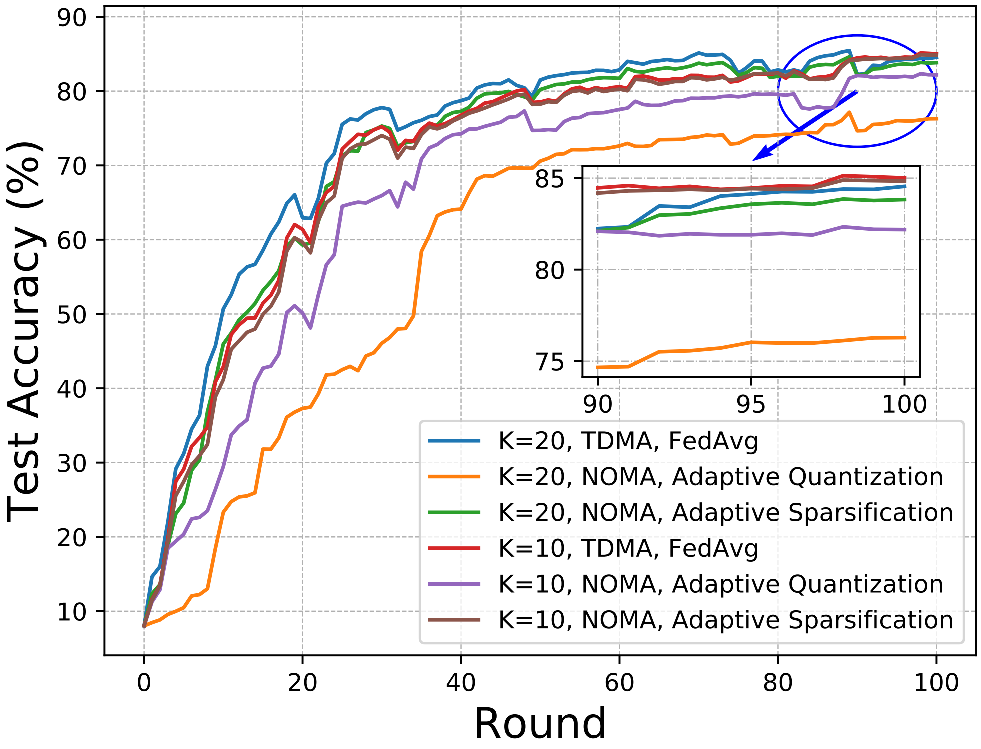


Fig. 2.2: Test accuracy of MNIST under non-IID vs. round

In Fig. 2.2, the test results for TDMA FedAvg, NOMA with adaptive quantization, and NOMA with adaptive sparsification for $K = 10$ and $K = 20$ are presented. The average compression rate of adaptive compression to measure the compression effect. It is summarized in Table 2.2. When the number of clients participating in the uplink process is fixed, adaptive quantization and adaptive sparsification achieve similar compression rates. Compared to $K = 10$, $K = 20$ requires a higher compression rate.

Table 2.2: Average Compression Rate

Uplink User Number (K)	$K = 10$	$K = 20$
Adaptive Quantization	1.82	3.03
Adaptive Sparsification	1.89	3.23

From Fig. 2.2, all schemes achieve similar test accuracy (over 80% except NOMA with adaptive quantization under $K = 20$ after 100 training rounds. TDMA FedAvg can consistently achieve better test accuracy than NOMA compression with $k = 10$ or $K = 20$. This indicates that compression affects the learning process. For TDMA FedAvg, $k = 20$ achieves better results since more users can reduce non-IID effects. However, for NOMA with compression, $K = 10$ achieves better results. When $K = 20$, mutual interference in NOMA causes a significant degradation in the data rate. It requires a higher compression rate, which causes destructive effects. Learning performance is better with adaptive sparsification than quantization in all NOMA scenarios.

In Fig. 2.3, the test accuracy of different schemes is evaluated from a time perspective. For simplicity, we only show the user number $K = 10$ here, but the conclusion also applies to $K = 20$ scenarios. Finishing an FL round takes $Kt_k + T_d$ for TDMA FedAvg, while $t_k + T_d$ for NOMA with adaptive compression schemes. It can be readily found that NOMA with adaptive compression can achieve 85% of test accuracy with around 70 s. However, it takes more than 500 seconds for TDMA FedAvg to achieve the same accuracy. NOMA-aided FL can save 7× communication time in FL model training. Notably, adaptive sparsification also achieves better learning performance than adaptive quantization during the whole learning

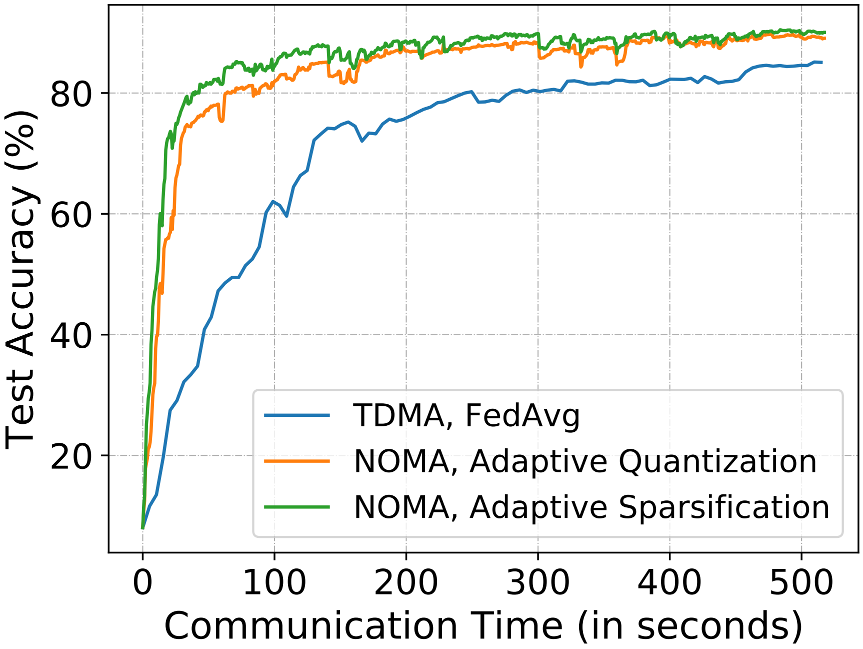
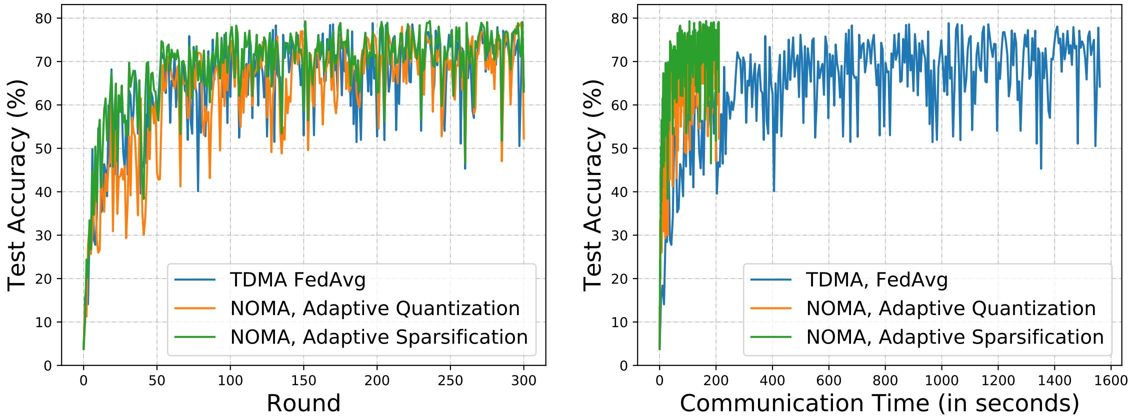


Fig. 2.3: Test accuracy of MNIST under non-IID vs. time

process in NOMA.

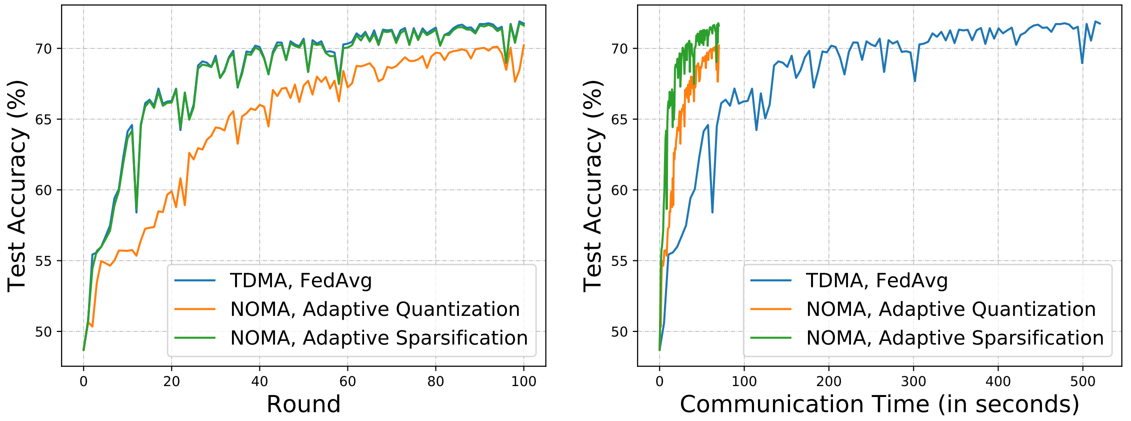
To generalize the proposed scheme, we run tests on another image dataset, FEMNIST, and a text dataset, Sent140. Here, we present the results for the scenarios $K = 20$. Figs. 2.4 and 2.5 show a fluctuation of test accuracy high, especially for the FEMNIST dataset under all scenarios. It is highly non-IDD, and each client has a distinct data distribution. We can see similar performance trends for TDMA FedAvg and our proposed NOMA with adaptive compression schemes. The TDMA FedAvg can consistently achieve the best performance. Sparsification can achieve better learning accuracy than quantization when NOMA is used. Notice that the TDMA FedAvg and NOMA with adaptive quantization achieve almost identical results from the round perspective. Therefore, it can be hard to distinguish the figures.

From the communication time perspective, to achieve 79.5% of the test accuracy in FEMNIST, TDMA FedAvg takes approximately 1600 s. Under NOMA with adaptive compression, it costs around 200 s. Similarly, the communication time of the Sent140 dataset is 510 s versus 75 s to achieve the 73.5% test accuracy. Our proposed NOMA with adaptive compression schemes can significantly save communication time and speed up the FL learning process.



(a) Round (b) Communication Time

Fig. 2.4: Test accuracy of FEMNIST



(a) Round (b) Communication Time

Fig. 2.5: Test accuracy of Sent140

2.5 Chapter Conclusion

In this chapter, we propose to apply NOMA in the FL for transmission of the client model to the server on the uplink. Practical fading channels are considered in the uplink.

Adaptive gradient compressions, including sparsification and quantization, are considered to satisfy the channel capacity constraints. The effectiveness of the proposed scheme is demonstrated by testing three public datasets with TDMA FedAvg as a baseline. From the communication time perspective, the proposed NOMA with adaptive compression can significantly reduce communication time while preserving test accuracy. Since FL is a distributed learning system, NOMA with power control or multiple antenna systems can further improve the learning performance in future research.

CHAPTER 3

Scheduling Policy and Power Allocation for Federated Learning in NOMA-Based MEC

3.1 Introduction

In the previous chapter, we applied NOMA with adaptive gradient compression to transmit the parameters of the FL model. The fading channel is considered in the uplink. All clients employ a uniform transmission power. We also applied a user scheduling scheme that considered the user's fairness and time budget. Great learning performance is achieved on three different datasets.

This chapter presents a joint user scheduling and power allocation scheme in the NOMA uplink for FL model transmission. A graph-based user scheduling scheme helps to achieve user fairness. And an optimal power allocation is constructed.

In communication-constrained environments, especially wireless networks, FL only selects a subset of connected clients for model transmission on the uplink. Since the clients collect the data from local environments, the data tends to be heterogeneous across clients. The user scheduling selects clients participating in the model parameter transmission process, contributing to global model aggregation. Therefore, user scheduling plays a crucial role in the contribution of the global model. In [21], three different user scheduling policies, i.e., random scheduling, round robin, and proportional fair, are proposed to schedule clients randomly, in groups with a round-robin way, or according to channel conditions, respectively. The convergence of three scheduling policies is analyzed. Running FL with a proportional fair scheme can reduce communication time compared to random and round-robin scheduling to achieve the same learning performance under high SINR. However, the researchers ignored the effects of data distributions. A coordinated scheduling and power control scheme in cloud radio access networks is introduced in [48]. A maximum weighted sum-rate problem is formulated and transformed into a maximum-weight clique problem to achieve the system's weighted sum data rate. The user scheduling problem is then solved using a graph-theoretical approach. The power allocation problem is resolved using MAPEL [49] to achieve maximum weighted throughput through power control. In [50], the spectrum efficient resource management problem (SERMP) under NOMA is investigated. It is transformed into a maximum-weight independent set problem and solved using graph theory. Jointly considering channel assignment and power control helps achieve high spectrum efficiency.

Motivated by the above-mentioned user scheduling and power allocation scheme, we apply user scheduling and power allocation in FL with NOMA transmission and adaptive gradient compression. As implied in Chapter 2, channel capacity can significantly affect FL learning performance. The user scheduling problem is first formulated to maximize the weighted sum rate. This would allow more information to be transmitted to obtain better learning results. Then, the maximum weighted sum-rate problem is transformed into a maximum weight independent set problem, which is solved with graph theory. The power allocation scheme is applied to maximize performance in each round.

3.2 System Model

As described in Chapter 2, the transmission of the FL model parameters in the uplink is transmitted using NOMA. The wireless links between the server and clients are assumed to be fading channels. Adaptive gradient quantization is applied here to satisfy the channel capacity constraints. Specifically, we assume that the total number of clients in the FL system is M, and the maximum number of clients selected to participate in NOMA uplink transmission is K. The number of iterations or rounds that the learning model needs to converge is T. Let $\mathcal{M}$, $\mathcal{K}$, and $\mathcal{T}$ be the set of all clients, the clients participating in the NOMA uplink and FL training rounds, respectively. Due to bandwidth limitations, the number of clients allowed to participate in the NOMA uplink in each round is usually much smaller than the total number of clients, that is, $M \gg K$. The number of rounds required to converge the FL model is generally small. We assume $M \geq K \times T$ here.

3.2.1 Problem Formulation

As mentioned in Chapter 2, we consider user fairness in scheduling. Given the massive number of clients and the stake of fairness, we assume each client can be selected to participate in the whole learning process at most once. The problem is formulated as an optimization problem that maximizes the weighted sum rate of all participants. Three following constraints are considered:

- $C1$: Each client can be selected at most once during the learning process.
- $C2$: At most K clients can be selected to participate in the NOMA uplink in each FL round.
- $C3$: The transmission power of each client is limited.

Then, the optimization problem is formulated as follows:

$$\max \sum_{m,t} \tau_m^t \Lambda_m^t R_m^t \tag{3.1a}$$

$$s.t. \quad \sum_{t} \Lambda_m^t \leq 1, \forall m, \tag{3.1b}$$

$$\sum_{m} \Lambda_m^t \leq K, \forall t, \tag{3.1c}$$

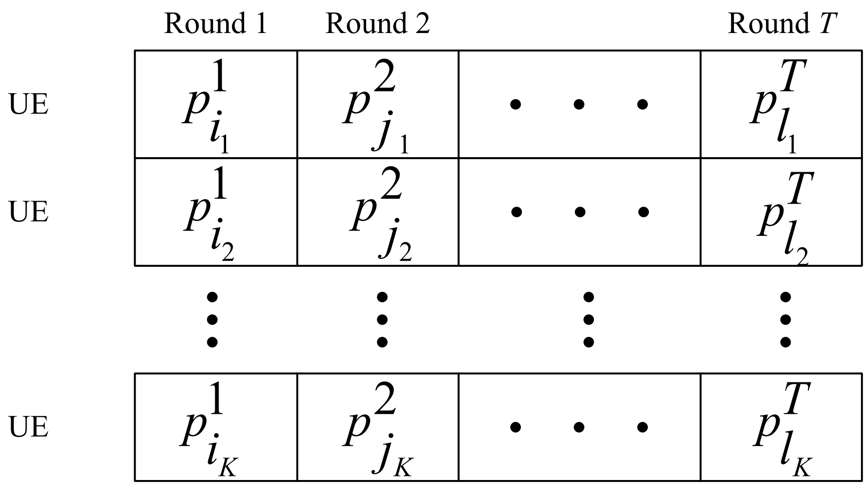
$$0 \leq p_m^t \leq p_m^{t\max}, \forall (m,t) \in \mathcal{M} \times \mathcal{T}, \tag{3.1d}$$

$$\Lambda_m^t \in \{0,1\}, \forall (m,t) \in \mathcal{M} \times \mathcal{T}, \tag{3.1e}$$

where R_m^t is the data rate for client m at round t, and τ_m^t is the weight for the data rate. $\Lambda_m^t \in 0,1$ is a binary variable that determines whether client m is selected in rount t. If $\Lambda_m^t = 1$, client m is selected to participate in the NOMA uplink in the round t. Otherwise, $\Lambda_m^t = 0$. Therefore, the constraint (3.1b), (3.1c), (3.1d) corresponds to $C1$, $C2$, and $C3$, respectively. All possible scheduling patterns need to be traversed to find the optimal user scheduling scheme to achieve the maximum weighted sum rate under these constraints. This is highly complex when the total number of clients is large, and the number of clients allowed to participate in the NOMA uplink in each round is small. Toward that end, a graph-theory-based maximum-weight independent set problem is formulated and solved. Once the user scheduling pattern is specified, the power allocation problem is solved using MAPEL [49].

3.3 User Scheduling and Power Allocation

First, the user scheduling and power allocation diagram are given in Fig. 3.1. There are total T columns and K rows. Each column represents an FL round, and at most K users are in each column. Here, i, j, l are used to represent clients. And $(i_1, i_2, \ldots, i_K)$, $(j_1, j_2, \ldots, j_K)$, $\cdots$, and $(l_1, l_2, \ldots, l_K)$ are different combinations of users with K clients in each set in the FL round. p_k^t is the transmission power of the user k at round t. Therefore, Fig. 3.1 overviews user scheduling and power allocation for FL NOMA uplink transmission.

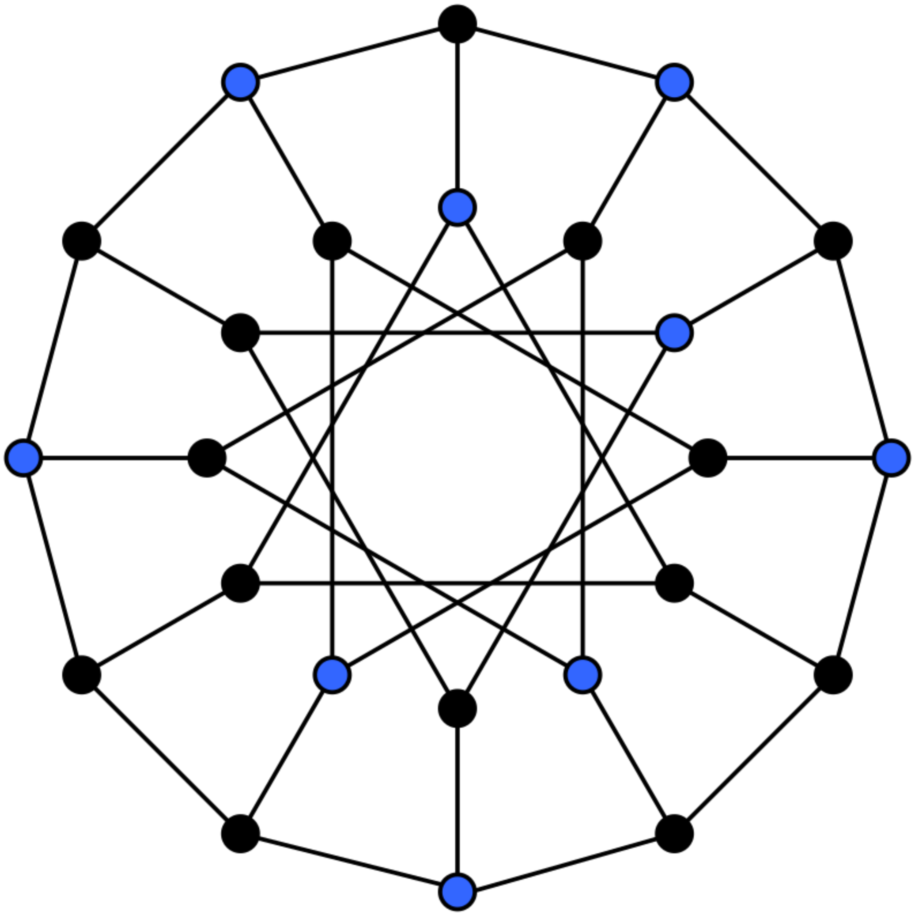


	Round 1	Round 2		Round T
UE	$p_{i_1}^1$	$p_{j_1}^2$	• • •	$p_{l_1}^T$
UE	$p_{i_2}^1$	$p_{j_2}^2$	• • •	$p_{l_2}^T$
	⋮	⋮	⋮	⋮
UE	$p_{i_K}^1$	$p_{j_K}^2$	• • •	$p_{l_K}^T$

Fig. 3.1: Diagram of user scheduling and power allocation

We first review all possible user patterns for the proposed joint user scheduling and power allocation scheme. Then, optimal power allocation is applied to each pattern to find the optimal one. This is very difficult to solve. The user scheduling aims to maximize the system's weighted sum rate. We first transform the maximum weighted sum rate problem into a maximum-weight independent set problem and solve it using a graph-theory-based method.

We first introduce the maximum-weight independent set problem. An independent set is a subgraph of an undirected graph with no edge between any two vertices. The maximum independent set is the independent set with the most nodes in a given graph G. Fig. 3.2 gives a maximum independent set with nine blue vertices in a generalized Petersen graph $G(12, 4)$ [51].



[51]

Fig. 3.2: Maximum independent set

3.3.1 Scheduling Graph Construction

This section will describe how the user scheduling graph is constructed and how the maximum weighted sum rate problem is transformed into the maximum weight independent set problem. In the following context, we will use vertex and edge concepts from graph theory.

Let $\mathcal{S}$ be the set that includes all possible scheduling patterns for all clients and FL rounds. $s \in \mathcal{S}$ is a possible schedule. The objective of the problem (3.1a) is to find a s that maximizes the weighted sum rate.

The scheduling graph is constructed as follows. First, the vertices are generated. Each vertex is a combination of multiple clients and a round. A vertex $v_j = (j_1, j_2, \ldots, j_K)t$ represents clients $j_1, j_2, \ldots, j_K$ scheduled in the round t. In this way, $\binom{M}{K} \times T$ vertices exist. Notice that a client j can appear in $\binom{M}{K} \times \frac{K}{M} \times T$ vertices, and each user combination $(j_1, j_2, \ldots, j_K)$ appears in T vertices. The edge construction should follow the constraints $C1$ and $C2$ mentioned above in the problem formulation part. Specifically, for two vertices $v_i = (i_1, i_2, \ldots, i_K)t_i$ and $v_j = (j_1, j_2, \ldots, j_K)t_j$, if $i_k \in \{j_1, j_2, \ldots, j_K\}, \forall k = \{1, \ldots, K\}$ (violates $C1$) or $t_i = t_j$ (violates $C2$), then v_i and v_j are connected with an edge. If the intersection of two vertices is non-empty, either from the client or round perspective, then the two vertices are connected. Then, when selecting vertices to construct an independent set, the constraints $C1$ and $C2$ are satisfied.

Here is an example of scheduling graphs with $M = 4, K = 1, T = 2$. There are total $\binom{4}{1} \times 2 = 8$ vertices. The edges are constructed as mentioned above. In Fig. 3.3, each vertex only connects to the vertices with intersections. For vertex (1)1, which means that client 1 is scheduled to round 1, the possible independent sets are $\{(1)1, (2)2\}, \{(1)1, (3)2\}, \{(1)1, (4)2\}$. Similarly, we can find all independent sets for each vertex in the graph.

When we set the weight of each vertex as the sum of the data rate of the clients scheduled to the round specified by the vertex, the sum of the weight of all vertices in an independent set equals the sum of the data rate of a possible user pattern. The maximum weighted sum rate problem is transformed into a maximum-weight independent set problem. The maximum-weight independent set problem involves searching all possible independent sets and finding the one independent set with the maximum weight value.

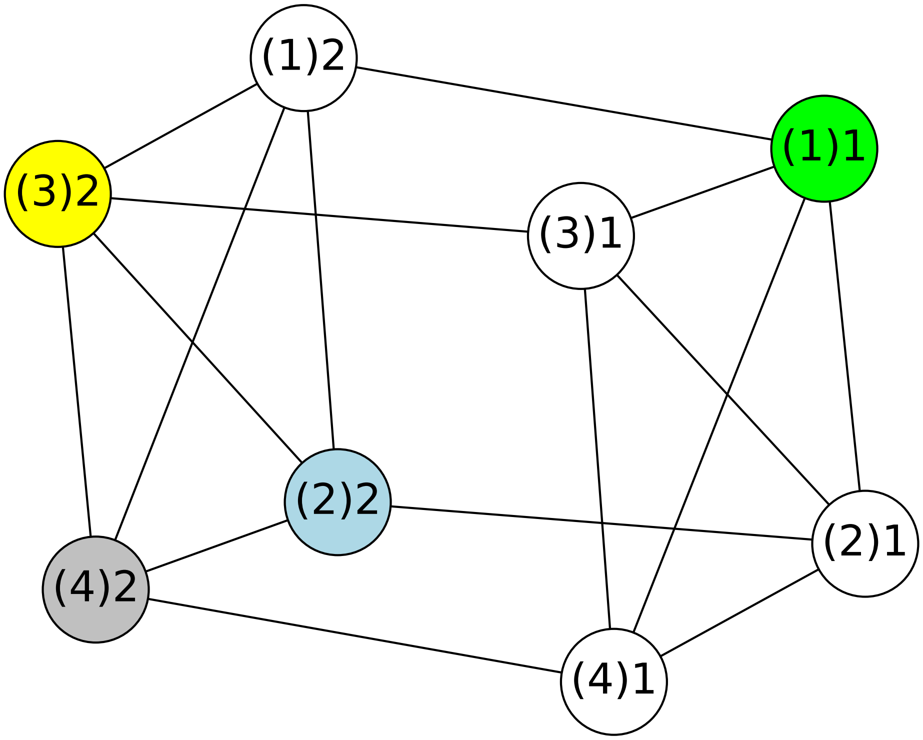


Fig. 3.3: A scheduling graph example

3.3.2 Optimal Scheduling Pattern

The scheduling graph can be extremely complicated when the number of clients M is large. It can be challenging to review all vertices and find all independent sets. The maximum-weight independent set problem is a strongly NP-hard problem [52], which is unlikely that an efficient algorithm exists to find the solution. Here, a greedy-based algorithm is employed to find the local optimal vertex and the next optimal vertex. This continues until the maximum independent set is achieved.

As mentioned above, we first calculate the weight of each vertex as the sum data rate of the clients scheduled in the round represented by the vertex. The weight of vertex v_j is defined as:

$$\tau(v_j) = \sum_{k \in v_j} \tau_k^t R_k^t, \tag{3.2}$$

where k is a client of the scheduled clients in round t. Then the sum of the weight of all vertices in an independent set, which equals the sum rate of a scheduling pattern, can be calculated as:

$$\sum_j \tau(v_j) = \sum_{k,t} \tau_k^t R_k^t, \forall (k,t) \in s, \tag{3.3}$$

where v_j represents vertex in an independent set s, and the summation involves all vertices in s.

The objective function (3.1a) is equal to maximizing 3.3, which is the maximum-weight independent set problem. The algorithm to find the optimal scheduling pattern is described in Algorithm 2.

Algorithm 2 Optimal Scheduling Selection

```
Require: $\mathcal{M}, \mathcal{K}, \mathcal{T}, p_m^t$, and $h_m^t$.
Initialize $\mathbf{O} = \emptyset$
Construct scheduling graph G
Compute $\tau(v), \forall v \in G$
while $G \neq \emptyset$ do
    $Q = \left\{ v | \tau(v) \geq \sum_{u \in J(v)} \frac{\tau(u)}{\beta(u)+1} \right\}$
    Select $v^* = \arg\max_{v \in Q} \frac{\tau(v)}{\beta(v)+1}$
    Set $\mathbf{O} = \mathbf{O} \cup \{v^*\}$
    Set $G = G - J(v^*)$
end while
Output $\mathbf{O}$
```

Here, $\mathbf{O}$ is the maximum-weight independent set, the optimal scheduling pattern to maximize the weighted sum rate. $J(v)$ is the subgraph of G that contains the vertex v and the neighboring vertices of v. And $\beta(v)$ is the degree of v, the number of vertices connected to v. Q is the set of vertices where the weight of the vertex v is greater than the average weight of $J(v)$. v^* is the local optimal vertex to maximize the average weight of $J(v)$. The algorithm 2 first calculates the weight of each vertex and then finds the local optimal vertex to maximize the independent set. The selected vertex with its adjacent vertices will be removed from the graph, and the next optimal vertex will be searched. In this way, the maximum-weight independent set can be solved.

3.3.3 Power Allocation

When the user scheduling pattern is determined, the power allocation can be executed based on the channel condition to achieve the maximum sum rate. We use MAPEL [49] to solve the power allocation problem in NOMA. The objective function (3.1a) as a logarithmic function of SINR increases monotonically. It can be transformed into multiplying a series of exponential linear fraction functions. Then, the optimal power allocation problem for a specified user scheduling pattern can be written as:

$$\max \prod_{k=1}^{K} (\frac{\mu_k(\mathbf{p})}{\phi_k(\mathbf{p})})^{\tau_k}, \tag{3.4a}$$

$$s.t. \quad 0 \leq p_k \leq p_k^{\max}, \forall k \in \mathcal{K}. \tag{3.4b}$$

where $\mathbf{p}$ is the power vector, $\mu_k(\mathbf{p}) = \sum_{j=k}^{K} p_j h_j^2 + \sigma^2$, and $\phi_k(\mathbf{p}) = \sum_{j=k+1}^{K} p_j h_j^2 + \sigma^2$. By letting $\mathbf{z}_k = \frac{\mu_k(\mathbf{p})}{\phi_k(\mathbf{p})}$, problem (3.4) can be reformulated as

$$\max \prod_{k=1}^{K} (\mathbf{z}_k)^{\tau_k} \tag{3.5a}$$

$$s.t. \quad 0 \leq \mathbf{z}_k \leq \frac{\mu_k(\mathbf{p})}{\phi_k(\mathbf{p})}, \forall k \in \mathcal{K}, \tag{3.5b}$$

$$0 \leq p_k \leq p_k^{\max}, \forall k \in \mathcal{K}. \tag{3.5c}$$

The function $f(\mathbf{x}) = \prod_{k=1}^{K} (x_k)^{\tau_k}$ is an increasing function for all positive x_k, where $\mathbf{x}$ is the set of e_k. For two vectors $\mathbf{x}_l$ and $\mathbf{x}_m$, $\mathbf{x}_l \succeq \mathbf{x}_m$ means that each element in $\mathbf{x}_l$ is greater than the element in the same index in $\mathbf{x}_m$. Then we have $f(\mathbf{x}_l) > f(\mathbf{x}_m)$. The optimal solution occurs when $\mathbf{z}_k^* = \frac{\mu_k(\mathbf{p})}{\phi_k(\mathbf{p})}$, and p_k is in the feasible set. Now, this problem can be regarded as a multiplicative linear fractional programming (MLFP) problem, and K linear equations can be written as:

$$z_k^* \phi_k(\mathbf{p}^*) - \mu_k(\mathbf{p}^*) = 0, \forall k \in \mathcal{K}. \tag{3.6}$$

There are K linear equations, with K unknown variables p_k. Random channel gains make the linear equations independent, implying a unique optimal power allocation $\mathbf{p}^*$. This can be solved efficiently with the algorithm in [49].

3.4 Simulation

As mentioned above, we also applied NOMA to the FL model parameter uplink transmission. Adaptive gradient compression is used to satisfy channel constraints. In Chapter 2, a uniform transmission power is applied for all clients. Here, we compare the performance of the following four schemes: 1) optimal joint user scheduling with power allocation (our proposed scheme), 2) optimal user scheduling with maximum power, 3) random scheduling with optimal power allocation, and 4) random scheduling with maximum power.

3.4.1 Simulation Settings

All the simulations are run on the MNIST dataset. A fully connected neural network, LeNet-300-100, with two hidden layers, is used. The first layer has 300 neurons, and the second has 100 neurons. Thus, the total number of model parameters, including bias, is $266,610$.

The communication parameter is listed below. The uplink bandwidth for NOMA is $B = 4$ MHz, the uplink transmission duration is 0.2s, the pass loss exponent is $\alpha = 3$, and the noise power density is $\sigma^2 = -174$ dBm/Hz. The scheduling graph construction requires high memory to keep all information on the vertices and edges. So we set the number of clients $M = 300$ and the number of clients participating in the NOMA uplink $K = 3$. The maximum transmission power of any client is $p^{\max} = 0.01$ watts. Clients are uniformly distributed in a disk area with a radius of 500 m. Broadcast communication is applied to the downlink from the server to the client with no gradient compression. The downlink transmission time is $T_d = \max_k \frac{G}{B_d \log_2(1+p_d\gamma_k)}$, where G is the total bit length of the model, $B_d = 10$ MHz is the downlink bandwidth. $p_d = 0.2$ watts is the server transmission power, γ_k is the SINR from the server to the client k.

Learning hyperparameters are given in Table 3.1. The FL round T is also set to 35 rather than 100 to meet the computation memory constraints. To verify the effectiveness and robustness of the proposed scheme, the data is allocated non-IID across clients.

Table 3.1: Hyperparameters

Learning rate size (η)	**Batch size** ($\mathcal{B}$)	**FL Round** (T)	**Training set size**	**Testing set size**
0.01	10	35	90%	10%

3.4.2 Simulation Results

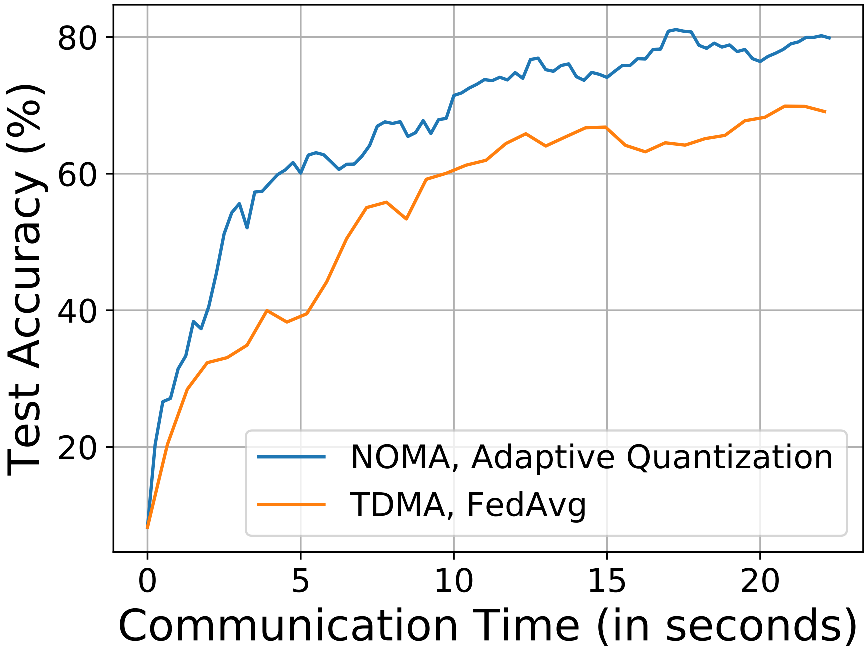


Fig. 3.4: Test accuracy vs time

Under NOMA uplink transmission with adaptive gradient compression, FL achieves better learning performance than TDMA FedAvg. With the specific settings mentioned above, Fig. 3.4 shows that FL under NOMA with adaptive compression achieves 70% of test accuracy after 10 s, while under TDMA, it takes about 22 s to achieve similar accuracy.

Fig. 3.5 compares four different user scheduling and power allocation schemes. It can be observed that all schemes except the random scheduling with maximum transmission power can obtain test accuracy above 60% after 35 learning rounds. This means that user scheduling and power allocation are crucial to improving learning performance. The optimal

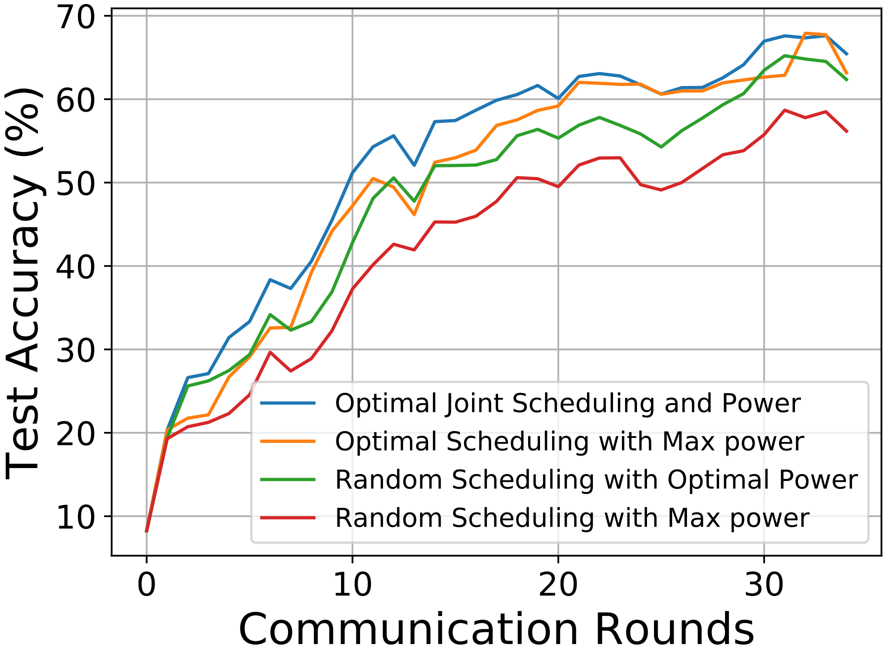


Fig. 3.5: Test accuracy vs round

joint scheduling and power allocation scheme consistently achieves the best performance during the learning process. Compared between user scheduling and power allocation, optimal scheduling with maximum transmission power achieves better learning performance than random scheduling with optimal power scheme, which implies that user scheduling plays a more critical role.

3.5 Chapter Conclusion

In this chapter, we jointly consider user scheduling and power allocation in NOMA uplink for FL model parameter transmission. The maximum weighted sum rate problem is transformed into a maximum-weight independent set problem, solved using a graph-theory-based method. User scheduling and power allocation can significantly affect learning performance.

CHAPTER 4

Approximate Wireless Communication for Lossy Gradient Updates in Federated Learning

4.1 Introduction

In previous chapters, we applied user scheduling and power allocation to NOMA uplink transmission of FL model parameters. Optimal joint user scheduling and power allocation can significantly improve FL learning performance. The simulation results also imply that user scheduling has a greater impact on learning performance than power allocation. These protocols can significantly reduce communication time and improve learning speed. However, reliable transmission is required. The information is transmitted with forward error correction (FEC) schemes [53] to ensure that every bit received is identical to the transmitted one. In addition, packet retransmission [54] is applied to recover the entire packet when the error is beyond the error correction capabilities of FEC. The adaptive gradient compression in the previous chapters allows the transmission in capacity-limited channels while maintaining good learning results. In another way, it indicates the error resiliency of ML systems. This motivated us to transmit the model parameters in an "approximate" way without FEC and packet retransmission to reduce the communication overhead.

4.1.1 Error Correction in Wireless Transmission

When UAVs or IoT devices serve as clients, the connection to the server is usually through wireless links [55]. Wireless channels are unreliable, resulting in huge errors during transmission. Several methods have been applied to overcome transmission errors in wireless communications. Wireless networks also use TCP/IP protocols to connect devices. In physical layers, transmission power can be increased to improve signal quality and signal-to-noise ratio (SNR) to mitigate noise impacts. However, this approach increases energy costs and drains the battery of energy-constrained mobile devices. Additionally, it could cause interference to adjacent clients. In the upper layers, the parity check, the checksum check, and the cyclic redundancy check (CRC) can be used for error detection. When errors are detected, forward error correction (FEC) [56] can be used to correct transmission errors. FEC encodes the original message into error correction codes (ECCs), adding redundant information. Therefore, the receiver can recover the transmitted message from the received message with errors. The simplest ECC is the repetition code, where each bit is repeated several times. The receiver recovers the message that occurs most often. However, it is not efficient because of the structure. Efficient ECCs, including low-density parity check (LDPC) and turbo code, are promising to achieve Shannon capacity. Although FEC can correct random errors, it introduces redundancy and requires additional computation and communication [54]. Additionally, redundancy determines error correction capability. That is, a good error correction capability requires high redundancy. And high redundancy resulted in high computation/communication costs. Therefore, there exists a trade-off between error correction capability and computational / communication efficiency. When the errors are beyond the ECC's correction capability, packet retransmission will be applied in the transport layer to ensure reliable transmission.

In FL, [57] focused on the transmission errors of the model. A wireless link with erroneous communication is considered. Rather than applying FEC or packet retransmission, the authors discard erroneous local models at the server and reuse the previous local models to continue training. This saves transmission time and ensures a reliable transmission. However, the most recent local models are lost and the global model converges more slowly. In [58], a FedLC framework was proposed. The User Datagram Protocol (UDP) rather than the Transmission Control Protocol (TCP) is applied to transmit the model update in a lossy communication channel. Data throughput is improved. FEC and packet retransmission are further employed to alleviate packet loss. Exploring FL model transmission in unreliable links is still required.

4.1.2 Approximate Communication

Modern ML systems are compromised by millions or even billions of model parameters.

The existing dropout randomly drops out nodes during training to reduce overfitting and improve generalization. This is one piece of evidence that ML is error-resilient. In [59], the error and fault tolerance in the ML system is reviewed. Large model designs in neural networks can be regarded as redundancy. The more neurons in the hidden layer, the better the error tolerance. In [60], the error resilience of the ML system is leveraged to design energy-efficient accelerators. Fewer bits are used for inexact computing. The FL system is a distributed ML system with model aggregation to obtain global models. It comes with the ability to overcome errors. In addition, model averaging helps to reduce error levels, especially when the number of clients is significant.

Approximate communication has been proposed to reduce communication bottlenecks in large-scale distributed systems [61]. This allows transmission with minor errors to achieve efficiency gains. Transmission accuracy is sacrificed to trade efficiency. There are two requirements for approximate communication: 1) the system has error resiliency, and 2) the error is acceptable. As a distributed ML system, FL shows error resiliency characteristics in multiple aspects. The gradients in ML vary in a small range to keep the model stable. This is demonstrated in empirical studies in [42, 62, 63]. The gradients are distributed in a small range within $(-1, 1)$ or even $(-0.01, 0.01)$. In [42], the gradient histograms in fully connected layers and convolutional layers are presented in different training iterations. The probability density function (PDF) and cumulative distribution functions (CDFs) of the gradients are given in [62]. The gradients are Gaussian or near Gaussian distributed. Since the gradient value is small, FL performance would not be affected when the transmission error is constrained to an acceptable level. The error resilience of the FL system can also be indicated by gradient compression. In previous chapters, we applied gradient quantization and sparsification to compress the gradient vector size and individual gradient bit representation. Learning performance is achieved with a slight loss due to information loss. Lastly, the FL aggregation process is a weighted averaging. The average operation helps to constrain the error to an acceptable level.

4.1.3 Literature Review

Approximate communication has been applied in various domains, such as network-on-chip (NoC) design and media transmission. The principle of applying approximate communication to the NoC design is to improve energy efficiency with less computation. In [64], approximate communication was used to design the photonic NoC to reduce the overhead laser and turning power. A 56.4% reduction in power consumption is obtained. In [65] a quality control method is introduced that reduces the size of the packet and reduces power consumption and transmission latency. Quality requirements determine the error resilience levels and error threshold. In [66], packet production and reduction of errors for NoC are proposed as approximate communication solutions. For media transmissions, such as images or video, errors are allowed if the error level is acceptable for human eyes. The approximate communication was introduced in [67] to reduce the usage of spectroscopy in wireless media delivery. The significant bits in the packets are put in protected positions while the insignificant bits are approximately transmitted. Video quality is improved by 5 to 20 dB. In [68], approximate communication is used in distributed optimization with a Newton-type method. These works demonstrate the broad application of approximate communication to reduce communication overhead and transmission latency.

4.1.4 Contribution and Organization

Motivated by the work mentioned above, we introduce an approximate communication framework to reduce latency in FL training and speed up the learning process. The FL model parameters are transmitted with errors caused by channel noise. No FEC or packet retransmission is executed on the transmitter side to help correct errors on the receiver side. Sacrificing accuracy provides low latency, reduced communication overhead, and decreased computation.

The principal contributions can be summarized as follows:

- A mathematical analysis of ML gradient is given. The backpropagation could result in gradient vanishing and gradient exploding problems.

- Prior knowledge of the gradients is utilized to confine the received gradients in a small range. The errors are constrained to a small range, making them acceptable to the FL system.

- The gradients are transmitted approximately without FEC and packet retransmission. Additionally, gray coding with high-order modulation protects the most significant bits.

- Extensive simulations are performed to show the effectiveness of the proposed approximate communication protocol for transmitting the FL model.

In Section 4.2, the system model is presented, including the FL model and the wireless channel model. Section 4.3 provides the mathematical analysis of gradients in fully connected networks and convolutional neural networks. The proposed method for FL model transmission is demonstrated in Section 4.4. The simulation settings and results are presented in Section 4.5. And finally, the chapter conclusions are given in Section 4.6.

4.2 System Model

4.2.1 FL System

The FL model is similar to the model in Chapter 1. M clients are connected to the server to train global models. Here, the gradients instead of weights information are uploaded from the client to the server. The local gradient at each client can be calculated as follows:

$$g_t^m = \nabla F_m(w_t). \tag{4.1}$$

Then, the global gradient after aggregation at the server is

$$g_t = \sum_{m=1}^{M} \frac{|D_m|}{|D|} g_t^m. \tag{4.2}$$

The server stores the global weight information of the previous round and calculates the current global model as follows:

$$w_{t+1} = w_t - \eta g_t. \tag{4.3}$$

The calculated global model is sent back to the clients for the next round of training.

4.2.2 Wireless Channel Model

FL operates at the application layer without knowing the details of implementing the lower-layer gradient transmission. It accepts data from the lower layers, assuming reliable transmission. Here, we consider a wireless transmission with the time division multiple access (TDMA) scheme. The received signal at the server can be written as:

$$r_t^m = \sqrt{p_t^m (d^m)^{-\alpha}} h_t^m g_t^m + n_t^m, \tag{4.4}$$

where r_t^m denotes the received signal from client m at round t. p is the transmission power, d is the transmission distance, α is the path loss exponent, h denotes the fading factor, g is the gradient, and n represents the Gaussian noise. $c_t^m = \sqrt{p_t^m (d^m)^{-\alpha}} h_t^m$ is the channel gain, which is assumed to be known by both the transmitter and the receiver. The noise resulted in transmission errors.

The quadrature amplitude modulation (QAM) scheme is applied. Gradients are converted into bits, then mapped to symbols, and transmitted through the wireless fading channels. The signal is decoded with maximum likelihood estimation on the receiver side and then demodulated with the QAM constellation. Then, the bits are transformed back to the gradients. The demodulation process can be represented as follows:

$$\hat{g}_t^m = \arg_{\bar{g}_t^m \in \mathcal{G}} \min ||r_t^m - \sqrt{p_t^m (d^m)^{-\alpha}} h_t^m \bar{g}_t^m||^2, \tag{4.5}$$

where $\mathcal{G}$ is the set of points on the constellation diagram. $\bar{g}_t^m$ denotes a possible symbol point within $\mathcal{G}$, while $\hat{g}_t^m$ denotes the optimal one.

The main notation used in the article is summarized in Table 4.1.

Table 4.1: Summary of Notations

Notation	**Definition**
M; m	The total number of clients connected to the server; the client index
L; l	The last layer of the neural network; the neural network layer index
$\boldsymbol{x}$; $\boldsymbol{y}$; $\hat{y}$	Features of a data point sample; corresponding true label of the data point; the predicted label
w; g; b	Model weight; model gradient; neural network bias
η; δ	Learning rate; intermediate quantity as "error"
C; $\sigma(\cdot)$	Loss function; activation function
z; a	Intermediate output of neuron; neuron output
N_l; N_g; N	The number of neurons in the l-th layer; the number of gradients in neural network; the number of data samples
i; j; k; p; q	Index
s; t; u, v	Index
t	Time (round) index
h; α; n; r	Channel factor; path loss exponent; noise; received signal
p_t^m; d^m	The transmission power of the client m at time t; Distance between the client m and the server

4.3 Gradient Analysis with Back-propagation

ML problems are optimization problems. ML model training aims to find the optimal point to minimize the loss functions. Since neural networks are usually non-convex, SGD and back-propagation are typical methods for model training. However, gradient vanishing or exploding problems can occur in deep neural networks, preventing model learning. The transmission of gradients with errors can cause the gradient to change in random directions. The received gradients can be substantial, making the model unstable. Or they can be tiny. And nothing is learned. Studying the gradient behavior and the existing method to prevent gradient vanishing/exploding helps to design the method to perform approximate communication.

4.3.1 Gradient in Fully Connected Neural Networks

In neural networks, the neuron is the fundamental unit for processing input and delivering the information to the next layer [69]. The diagram is illustrated in Fig. 4.1. It shows the j-th neuron in the l-th layer.

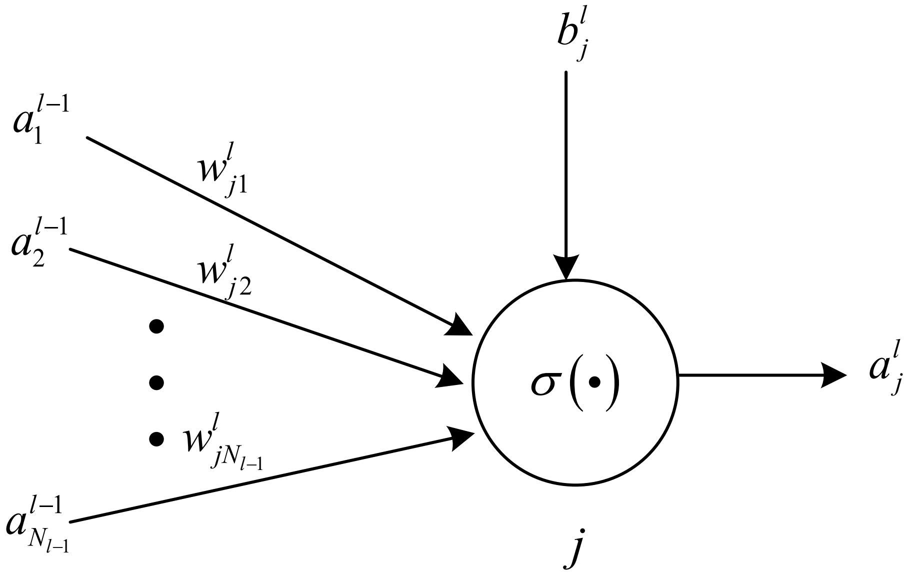


Fig. 4.1: Artificial neuron diagram

The feedforward equation for this neuron can be written as

$$\begin{aligned} z_j^l &= b_j^l + \sum_k^{N_{l-1}} w_{jk}^l a_k^{l-1}, \\ a_j^l &= \sigma(z_j^l). \end{aligned} \tag{4.6}$$

The activation function $\sigma(\cdot)$ provides a non-linear generalization ability to the model. N_{l-1} is the number of neurons in the $(l-1)$-th layer, which is the adjacent layer in front of the l-th layer. a is the output of the activation function. The neuron accepts data inputs when it is located in the first layer. And equation (4.6) becomes

$$\begin{aligned} z_j^1 &= b_j^1 + \sum_k^{N_0} w_{jk}^1 x_k^0, \\ a_j^1 &= \sigma(z_j^1). \end{aligned} \tag{4.7}$$

Since backpropagation is applied in the model optimization, the four fundamental equations for a fully connected neural network are [70]:

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L), \tag{4.8a}$$

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k^{N_{l+1}} \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial a_j^l}\frac{\partial a_j^l}{\partial z_j^l} = \sum_k^{N_{l+1}} \delta_k^{l+1} w_{kj}^{l+1}\sigma'(z_j^l), \tag{4.8b}$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l, \tag{4.8c}$$

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}. \tag{4.8d}$$

L is the final layer in neural networks. δ_j^l represents the "error". It is introduced as an intermediate quantity for the calculation of partial derivatives in Equations (4.8c) and (4.8d). Then the weight update can be written as

$$w_{jk}^l = w_{jk}^l - \eta\delta_j^l a_k^{l-1}. \tag{4.9}$$

Calculating the gradient $g_{jk}^l = \frac{\partial C}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}$ involves two terms δ_j^l and a_k^{l-1}. The output of the activation function a_k^{l-1} depends on the selection of the activation function. For the sigmoid function, $a_k^{l-1} \in (0,1)$ regardless of the input value z_k^{l-1}. A more detailed analysis of the activation function can be found in [71].

Another term δ_j^l is calculated in equation (4.8b). It is a summation of the products of three terms, error in the next layer δ_k^{l+1}, weight w_{kj}^{l+1}, and derivative of the activation function $\sigma'(z_j^l)$. The summation runs through all neurons in the $(l+1)$-th layer.

First, the value of $\sigma'(z_j^l)$ depends on the specific activation function. It ranges from $(0, 0.25)$ for the sigmoid function and $\{0,1\}$ for ReLU. The weight value w_{kj}^{l+1} depends on the initialization of the model, the learning rate η, and the previous round gradients based on equation (4.3). The model initialization usually generates the initial weight randomly within the range of $(-1,1)$ [72] from Gaussian or uniform distribution [73]. The recent initialization method improves initialization. In [74], the weight is initialized with a uniform distribution considering the number of neurons, $w \sim U[\frac{1}{\sqrt{N_p}}, \frac{1}{\sqrt{N_p}}]$, N_p is the number of neurons in the previous layer. The weight is initialized with a Gaussian distribution in [75]. The $w \sim \mathcal{N}(0, \sqrt{(2/N_p)})$, where 99.7% of the weights are within $[3 * \sqrt{(2/N_p)}, 3 * \sqrt{(2/N_p)}]$. Lastly, the error δ_k^{l+1} can be expressed as in Equation (4.8b) with elements in the $(l+2)$-th layer as follows:

$$\begin{aligned} \delta_k^{l+1} = \frac{\partial C}{\partial z_k^{l+1}} &= \sum_i^{N_{l+2}} \frac{\partial C}{\partial z_i^{l+2}} \frac{\partial z_i^{l+2}}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial z_k^{l+1}} \\ &= \sum_i^{N_{l+2}} \delta_i^{l+2} w_{ik}^{l+2} \sigma'(z_k^{l+1}). \end{aligned} \tag{4.10}$$

The error in the previous layers is related to the subsequent layers, and the relationship ends at the final layer.

For classification problems, softmax is applied to normalize the prediction probability. Cross-entropy is the common loss function. It can be written as:

$$C = -\sum_i y_i \log(\hat{y}_i), \tag{4.11}$$

where y_i is the input label, $\hat{y}_i$ is the softmax probability, i.e.,

$$\hat{y}_i = \sigma_s(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}. \tag{4.12}$$

The derivative of $\hat{y}_i$ is

$$\frac{\partial \hat{y}_i}{\partial z_j} = \begin{cases} \hat{y}_i(1 - \hat{y}_j), & \text{if } i = j; \\ -\hat{y}_j \cdot \hat{y}_i, & \text{if } i \neq j. \end{cases} \tag{4.13}$$

Then, the equation (4.8a) can be written as:

$$\begin{aligned} \delta_j^L = \frac{\partial C}{\partial \hat{y}_i^L} \frac{\partial \hat{y}_i^L}{\partial z_j^L} &= -\sum_i y_i \frac{\partial \log(\hat{y}_i)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j} = -\sum_i y_i \frac{1}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j}, \\ &= -y_j(1 - \hat{y}_j) - \sum_{i \neq j} y_i \frac{1}{\hat{y}_i}(-\hat{y}_j \cdot \hat{y}_i), \\ &= \hat{y}_j \cdot \sum_i y_i - y_j. \end{aligned} \tag{4.14}$$

And, equation (4.8d) becomes:

$$\begin{aligned}\frac{\partial C}{\partial w^l_{jk}} &= \delta^l_j a^{l-1}_k \\ &= [\sum_p^{N_{l+1}} \delta^{l+1}_p w^{l+1}_{pj} \sigma'(z^l_j)] a^{l-1}_k \\ &= \{\sum_p^{N_{l+1}} [\sum_q^{N_{l+2}} \delta^{l+2}_q w^{l+2}_{qj} \sigma'(z^{l+1}_j)] w^{l+1}_{pj} \sigma'(z^l_j)\} a^{l-1}_k \\ &= \cdots \\ &= \sum_p^{N_{l+1}} \cdots [\sum_i^{N_L} \delta^L_i w^L_{ij} \sigma'(z^{L-1}_j)] \times \cdots \times a^{l-1}_k\end{aligned} \tag{4.15}$$

When $l = 1$, it becomes

$$\frac{\partial C}{\partial w^1_{jk}} = \sum_p^{N_{l+1}} \cdots [\sum_i^{N_L} \delta^L_i w^L_{ij} \sigma'(z^{L-1}_j)] \times \cdots \times x^0_k. \tag{4.16}$$

From equation (4.15), the gradient calculation involves multiplication and summation. When $\sum_p^{N_{l+1}} \delta^{l+1}_p w^{l+1}_{pj} \sigma'(z^l_j)$ for each layer is greater than 1, the multiplication increases the gradient even further. This could result in gradient problems. Conversely, when the summation is close to 0, gradient vanishing problems could occur.

4.3.2 Gradient in Convolutional Neural Networks

CNN is powerful in extracting local features from images. A typical CNN consists of compromised convolutional layers followed by grouping layers for feature learning and fully connected layers for classification [76]. Without loss of generality, we consider a CNN network with two convolutional layers, two max-pooling layers, and two fully connected layers. The diagram is shown in Fig. 4.2.

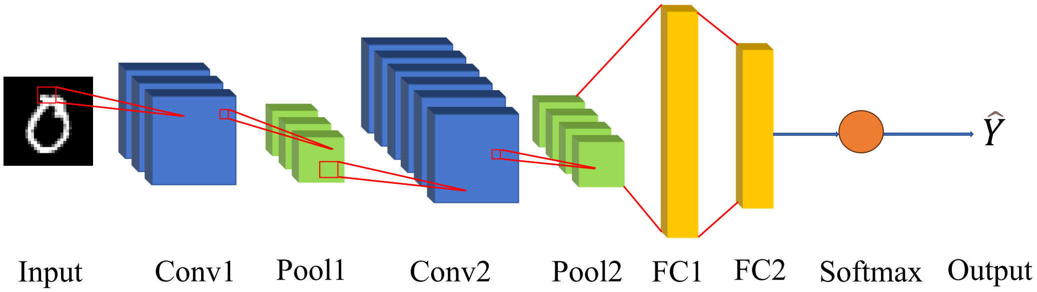


Fig. 4.2: Convolutional neural networks architecture

The feedforward process can be written as [77]:

$$z^1_{j,k} = b^1_{j,k} + \sum_p \sum_q w^1_{p,q} x^0_{j+p,k+q}, \tag{4.17a}$$

$$a^1_{j,k} = \sigma(z^1_{j,k}), \tag{4.17b}$$

$$a^2_{j,k} = \max(a^1_{2j,2k}, a^1_{2j+1,2k}, a^1_{2j,2k+1}, a^1_{2j+1,2k+1}), \tag{4.17c}$$

$$z^3_{j,k} = b^3_{j,k} + \sum_p \sum_q w^3_{p,q} a^2_{j+p,k+q}, \tag{4.17d}$$

$$a^3_{j,k} = \sigma(z^3_{j,k}), \tag{4.17e}$$

$$a^4_{j,k} = \max(a^3_{2j,2k}, a^3_{2j+1,2k}, a^3_{2j,2k+1}, a^3_{2j+1,2k+1}), \tag{4.17f}$$

$$z^5_i = b^5_i + \sum_{j,k} w^5_{i;j,k} a^4_{j,k}, \tag{4.17g}$$

$$a^5_i = \sigma(z^5_i), \tag{4.17h}$$

$$z^6_i = b^6_i + \sum_k w^6_{i,k} a^5_k, \tag{4.17i}$$

$$a^6_i = \sigma(z^6_i). \tag{4.17j}$$

Here, the 2 × 2 max-pooling layers follow the convolutional layer. The backpropagations for equation (4.17) are

$$\delta_i^6 = \frac{\partial C}{\partial z_i^6} = \frac{\partial C}{\partial a_i^6}\frac{\partial a_i^6}{\partial z_i^6} = \frac{\partial C}{\partial a_6^3}\sigma'(z_i^6), \tag{4.18a}$$

$$\begin{aligned}
\delta_i^5 &= \frac{\partial C}{\partial z_i^5} = \sum_k^{N_6} \frac{\partial C}{\partial z_k^6}\frac{\partial z_k^6}{\partial a_i^5}\frac{\partial a_i^5}{\partial z_i^5},\\
&= \sum_k^{N_6} \delta_k^6 w_{ki}^6 \sigma'(z_i^5),
\end{aligned} \tag{4.18b}$$

$$\begin{aligned}
\delta_{j,k}^3 &= \frac{\partial C}{\partial z_{j,k}^3} = \sum_i \frac{\partial C}{\partial z_i^5}\frac{\partial z_i^5}{\partial a_{s,t}^4}\frac{\partial a_{s,t}^4}{\partial z_{j,k}^3},\\
&= \sum_i \delta_i^5 w_{i;s,t}^5 \frac{\partial a_{s,t}^4}{\partial a_{j,k}^3}\frac{\partial a_{j,k}^3}{\partial z_{j,k}^3},\\
&= \sum_i \delta_i^5 w_{i;s,t}^5 \frac{\partial a_{s,t}^4}{\partial a_{j,k}^3}\sigma'(z_{j,k}^3),\\
&= \begin{cases} \sum_i \delta_i^5 w_{i;s,t}^5 \sigma'(z_{j,k}^3), & \text{if case 1;}\\ 0, & \text{otherwise;} \end{cases}
\end{aligned} \tag{4.18c}$$

$$\begin{aligned}
\delta_{j,k}^1 &= \frac{\partial C}{\partial z_{j,k}^1} = \sum_u \sum_v \frac{\partial C}{\partial z_{u,v}^3}\frac{\partial z_{u,v}^3}{\partial a_{s,t}^2}\frac{\partial a_{s,t}^2}{\partial z_{j,k}^1},\\
&= \sum_u \sum_v \delta_{u,v}^3 w_{u-s,v-t}^3 \frac{\partial a_{s,t}^2}{\partial a_{j,k}^1}\frac{\partial a_{j,k}^1}{\partial z_{j,k}^1},\\
&= \sum_u \sum_v \delta_{u,v}^3 w_{u-s,v-t}^3 \frac{\partial a_{s,t}^2}{\partial a_{j,k}^1}\sigma'(z_{j,k}^1),\\
&= \begin{cases} \sum_u \sum_v \delta_{u,v}^3 w_{u-s,v-t}^3 \sigma'(z_{j,k}^1), & \text{if case 2;}\\ 0, & \text{otherwise;} \end{cases}
\end{aligned} \tag{4.18d}$$

$$\frac{\partial C}{\partial w_{i,k}^6} = \frac{\partial C}{\partial z_i^6}\frac{\partial z_i^6}{\partial w_i, k^6} = \delta_i^6 a_k^5, \tag{4.18e}$$

$$\frac{\partial C}{\partial w_{i;j,k}^5} = \frac{\partial C}{\partial z_i^5}\frac{\partial z_i^5}{\partial w_{i;j,k}^5} = \delta_i^5 a_{j,k}^4, \tag{4.18f}$$

$$\frac{\partial C}{\partial w_{p,q}^3} = \frac{\partial C}{\partial z_{j,k}^3}\frac{\partial z_{j,k}^3}{\partial w_{p,q}^3} = \delta_{j,k}^3 a_{j+p,\ k+q}^2, \tag{4.18g}$$

$$\frac{\partial C}{\partial w_{p,q}^1} = \frac{\partial C}{\partial z_{j,k}^1}\frac{\partial z_{j,k}^1}{\partial w_{p,q}^1} = \delta_{j,k}^1 x_{j+p,\ k+q}^0. \tag{4.18h}$$

The softmax function is also applied in the final layer, and the cross-entropy function is used as the loss function. There are two types of gradients: $\frac{\partial C}{\partial w^6_{i,k}}$, $\frac{\partial C}{\partial w^5_{i;j,k}}$ in the fully connected layer, and $\frac{\partial C}{\partial w^3_{p,q}}$ and $\frac{\partial C}{\partial w^1_{p,q}}$ in the convolutional layer.

$\frac{\partial C}{\partial w^6_{i,k}}$ can be calculated from equation (4.18a). As mentioned above, it is identical to calculating in fully connected neural networks. For $\frac{\partial C}{\partial w^5_{i;j,k}}$,

$$\frac{\partial C}{\partial w^5_{i;j,k}} = [\sum_{k}^{N_6} \delta^6_k w^6_{ki} \sigma'(z^5_i)] a^4_{j,k}. \tag{4.19}$$

The summation counts all neurons in layer 6.

For $\frac{\partial C}{\partial w^3_{p,q}}$ and $\frac{\partial C}{\partial w^1_{p,q}}$, only the non-zero conditions are considered.

$$\begin{aligned} \frac{\partial C}{\partial w^3_{p,q}} &= \{\sum_i \delta^5_i w^5_{i;s,t} \sigma'(z^3_{j,k})\} a^2_{j+p,\ k+q} \\ &= \{\sum_i [\sum_k^{N_6} \delta^6_k w^6_{ki} \sigma'(z^5_i)] w^5_{i;s,t} \sigma'(z^3_{j,k})\} a^2_{j+p,\ k+q} \end{aligned} \tag{4.20}$$

$$\begin{aligned} \frac{\partial C}{\partial w^1_{p,q}} &= \{\sum_u \sum_v \delta^3_{u,v} w^3_{u-s,v-t} \sigma'(z^1_{j,k})\} x^0_{j+p,\ k+q} \\ &= \{\sum_u \sum_v [\sum_i \delta^5_i w^5_{i;s,t} \sigma'(z^3_{j,k})] w^3_{u-s,v-t} \sigma'(z^1_{j,k})\} \times x^0_{j+p,\ k+q} \\ &= \{\sum_u \sum_v [\sum_i (\sum_k^{N_6} \delta^6_k w^6_{ki} \sigma'(z^5_i)) w^5_{i;s,t} \sigma'(z^3_{j,k})] w^3_{u-s,v-t} \sigma'(z^1_{j,k})\} \times x^0_{j+p,\ k+q} \end{aligned} \tag{4.21}$$

The gradient calculation in convolutional layers also involves multiplication and summation, while the gradient in the front layers involves more arithmetic operations.

4.3.3 Gradient in Deep Neural Networks

When the neural networks get deeper, and the number of neurons in each layer increases, the summation and multiplication operations accumulate in backpropagation. This would result in gradient exploding problems [78]. The trained model becomes unstable and produces random results. It has been well-studied that gradient clipping, proper weight initialization, and batch normalization can prevent gradient exploding. Specifically, gradient clipping helps to constrain the gradient with a predefined threshold. The threshold is introduced as an additional hyperparameter. If the threshold is too small, the learning ability is restricted. If it is too large, the clipping effects may be affected.

When $\sum_p^{N_{l+1}} \delta_p^{l+1} w_{pj}^{l+1} \sigma'(z_j^l)$ is small in all layers, the gradient vanishing problem can occur [79]. The most effective method to mitigate the risk of gradient vanishing is using the ReLU activation function to replace the Sigmoid function. The ReLU function and its derivative is

$$\sigma(x) = \begin{cases} 0, & x \leq 0; \\ x, & x > 0. \end{cases} \tag{4.22}$$

$$\sigma'(x) = \begin{cases} 0, & x \leq 0; \\ 1, & x > 0. \end{cases} \tag{4.23}$$

This results in the derivative of the activation function in (4.8b) being 0 or 1. It contributes to the stabilization of the neural network.

4.4 Proposed Method

Motivated by the effectiveness of gradient clipping in mitigating gradient exploding problems, we first present a received bit masking scheme to constrain the received gradient values to a small range. Then, the error is also restricted to a small range. Approximate communication is applied to transmit the gradients with errors. No FEC or packet retransmission is used. The error level is measured with a l_2 norm. Small transmission errors produce better learning performance. Gray coding with high-order modulation protects the most significant bits to enhance learning performance. Finally, gradient compression is added to further reduce the communication costs in gradient transmission.

4.4.1 Received Bits Masking

In section 4.3, we provided the mathematical analysis of the gradient calculation in fully connected neural networks and CNN. Backpropagation can cause gradient exploding or vanishing problems. However, the gradient should be distributed in a small range in a good learning procedure. This section uses bit-masking to constrain the gradient in a small range.

In ML, gradients are usually represented with 32-bit floating-point numbers, defined by the IEEE-754 standard. The first bit is the sign bit, followed by 8 bits for the exponent part and 23 bits for the fraction part. In wireless transmission, an error can occur at any position. Interleaving can help disperse errors and avoid block corruption.

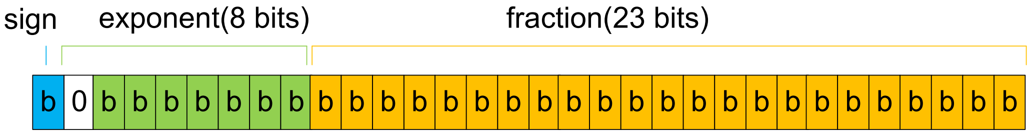


Fig. 4.3: Received gradient bit masking

As shown in Fig. 4.3, we mask the second bit, i.e., the first bit in the exponent part, of the received gradient as 0 at the receiver side. Other received bits are kept as is. This constrained the received gradient value in the $(-2, 2)$ range. And the magnitude of the transmission error is smaller than 2. The error level is restricted to an acceptable level.

4.4.2 Approximate Wireless Communication

Since the error is restricted to an acceptable level, the gradient can be transmitted without FEC and packet retransmission. The received gradients have errors and are not perfectly accurate. The error level is measured in [80] with a relative difference. Here, the relative error is defined as:

$$E_{rg} = \frac{|g - \tilde{g}|}{g}, \tag{4.24}$$

where g represents the original gradient value, $\tilde{g}$ denotes the received value after bit masking. Given the multitude of gradients present within the neural network, we employ the l_2-norm of errors instead of relative error to evaluate the impact of errors. The l_2-norm of errors is written as:

$$||E_r||_2 = \sqrt{\sum_{i=1}^{N_g} |g_i - \tilde{g}_i|^2}, \tag{4.25}$$

where N_g is the number of gradients in the neural network.

4.4.3 Most Significant Bit Protection

Furthermore, the gray coding can protect MSBs from erroneous wireless transmission. For example, when 16-QAM is employed with gray coding. The constellation map can be shown in Fig. 4.4.

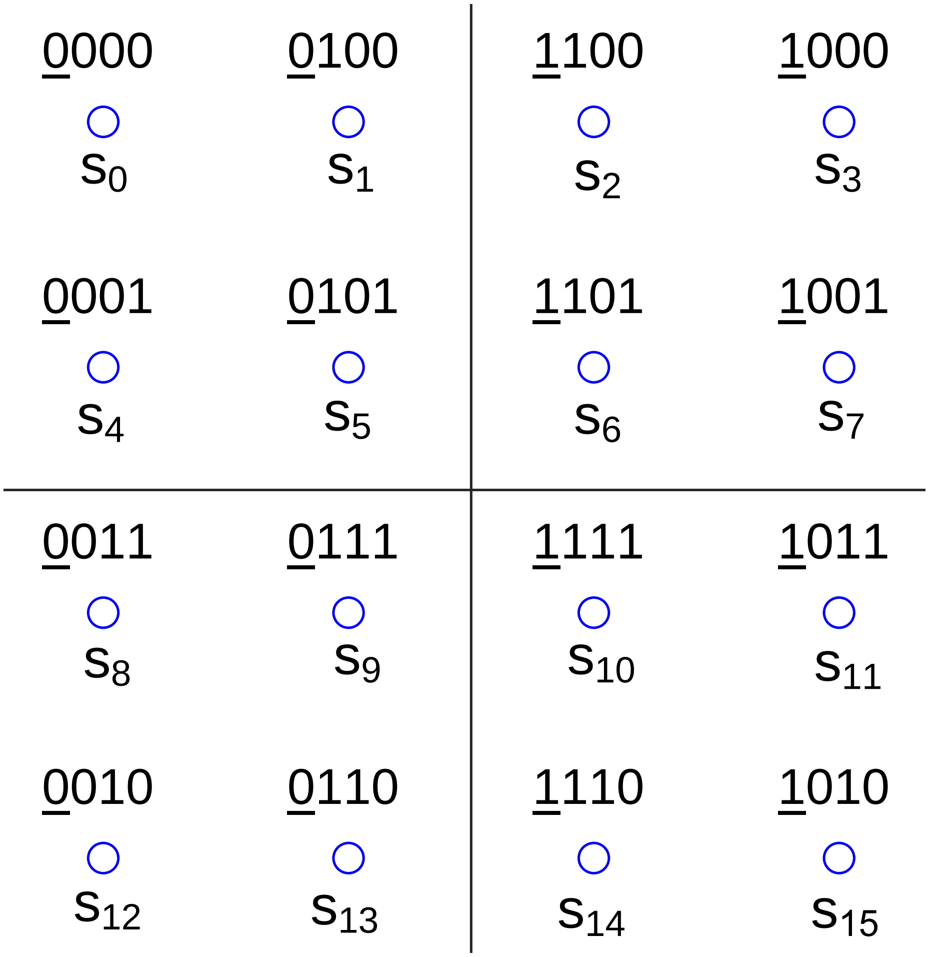


Fig. 4.4: Gray coding in 16-QAM constellation map

In Fig. 4.4, each symbol s represents 4 bits. The most left bit (underlined) represents the MSB, while the most right bit represents the least significant bit (LSB). Assume that the transmission probability for each symbol is identical. The error probability of MSB is smaller than that of LSB. For example, if symbol s_0 is transmitted and an error occurs due to noise. The decoded symbol can be s_1, s_4, or s_5 when the noise power is subordinate. For simplicity, we assume that the error symbol can only be s_1, s_4, s_5. The MSB remains constant, while the LSB changes when the decoded symbol is s_4 or s_5. The error appears twice. Assuming that the symbol error probability (SEP) of s_0 to s_1 is ρ, then the SEP for s_0 to s_4 is ρ and $\sqrt{2}\rho$ for s_5. This is summarized in Table 4.2.

Table 4.2: Gray Coding with 16-QAM MSB/LSB Error Count

Symbol	**Potential Error Symbol**	**MSB Error Count/ Probability**	**LSB Error Count/ Probability**
s_0	s_1, s_4, s_5	0 / 0	2 / $(1+\sqrt{2})\rho$
s_1	s_0, s_2, s_4, s_5, s_6	2 / $(1+\sqrt{2})\rho$	3 / $(1+2\sqrt{2})\rho$
s_4	s_0, s_1, s_5, s_8, s_9	0 / 0	2 / $(1+\sqrt{2})\rho$
s_5	$s_0, s_1, s_2, s_4,$ s_6, s_8, s_9, s_{10}	3 / $(1+2\sqrt{2})\rho$	3 / $(1+2\sqrt{2})\rho$

In the IEEE-754 standard, the exponent part determines the integer and fraction, while the fraction part only determines the fraction. The exponent part, located on the left, is more significant. Gray coding protects the exponent part, which reduces error levels.

4.4.4 Gradient Compression

As mentioned in the previous chapters, gradient compression can reduce communication costs. However, this would cause a slight loss in the final learning performance. Gradient sparsification is applied here. The gradients with the largest gradient magnitude and their position information are transmitted, specifically in an accurate method.

The entire process is summarized in Algorithm 3.

4.5 Simulation

We present extensive simulation results to verify our proposed methods. First, we

Algorithm 3 Approximate Wireless Communication for Federated Learning

```
Initialization: Server initializes $w_0$ and broadcasts to all Clients.
while not converge do
    Client:
        Receives the most recent aggregated global model.
        Performs local computation as Eq.(4.1).
        Sends local gradient directly without error correction coding through wireless channels.
    Server:
        Receives the local gradient with errors.
        Sets the second bit in the 32-bit gradient representation as 0, leaving the other bits as received.
        Sends the aggregated global model to clients.
end while
```

give the bit error rate (BER) versus the SNR under the specific wireless channel conditions. Lower-order modulation exhibits better BER at the same SNR level. Specifically, QPSK gets better BER than 16-QAM and 256-QAM. Then, we show the FL learning performance with the proposed method and the classical error correction and packet retransmission (ECRT) method. Three public image datasets are applied under different wireless conditions. Then, the learning performance is given under different SNRs. Better channel conditions can achieve better learning performance. Then, the error level is measured at a fixed SNR in various rounds and across SNR. Learning performance versus time is given to evaluate the time efficiency of the proposed method. Learning performance at different BERs is also shown. Finally, the result with added gradient compression is presented.

4.5.1 Simulation Settings

The simulation considers an FL system with $M = 100$ clients. Three different datasets are applied: MNIST, Fashion-MNIST, and Cifar. The IID and non-IID data distributions are considered. CNNs with varying architectures of the model are employed. CNN for the MNIST, Fashion-MNIST, and Cifar-10 datasets has 21,840, 65,558, and 62,006 model parameters, respectively. ReLU is used as an activation function. The learning rate η is 0.01, and the training batch size is 10.

For the communication model, the path loss exponent is $\alpha = 3$, and the distance between the server and the clients is 10 meters. Client transmission power is normalized to 1. Unless otherwise specified, QPSK modulation is applied in the simulation.

4.5.2 Simulation Results

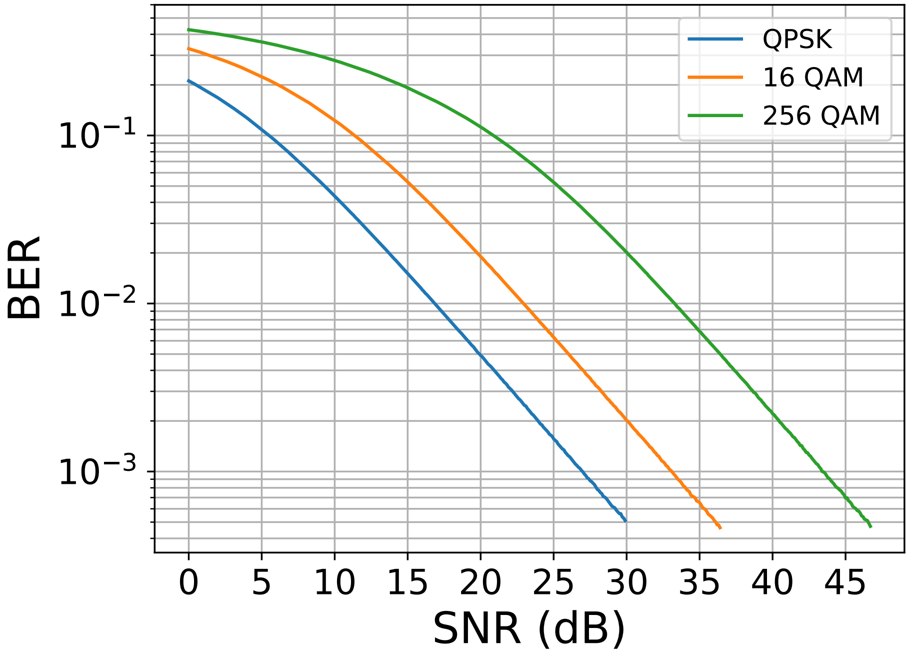


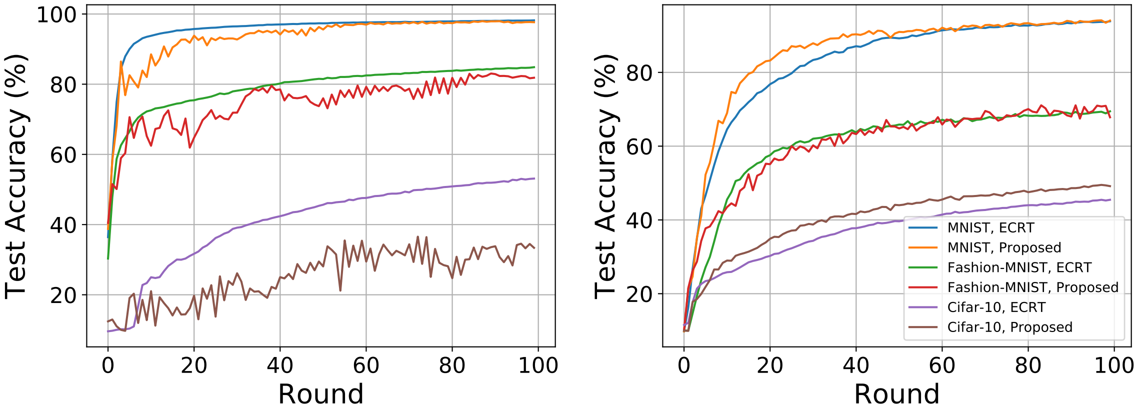
Fig. 4.5: BER

The BER results of QPSK, 16-QAM, and 256-QAM at different SNRs are given in Fig. 4.5. QPSK achieves the lowest BER. The BER at 0 dB, 10 dB, and 20 dB is summarized in Table 4.3.

Table 4.3: BER Summary

SNR	**QPSK**	**16-QAM**	**256-QAM**
0 dB	2.11×10^{-1}	3.28×10^{-1}	4.26×10^{-1}
10 dB	4.36×10^{-2}	1.23×10^{-1}	2.79×10^{-1}
20 dB	4.91×10^{-3}	1.90×10^{-2}	1.12×10^{-1}

First, the convergence of the proposed method is presented in Fig. 4.6. Compared with the ECRT scheme, the proposed scheme achieves similar convergence points under a high SNR regime while experiencing a marginal reduction under a low SNR regime. In Fig. 4.6(a), the learning performance with our proposed method experiences significant fluctuation compared to the transmission of ECRT. This variance is attributed to the transmission errors introduced by the approximate communication. Except for Cifar-10, the learning performance converges closely with ECRT transmission for MNIST and Fashion-MNIST. In a high SNR regime, that is, SNR = 20 dB, the fluctuation becomes smaller, as shown in Fig. 4.6(b). Owing to the non-IID data distribution, the proposed method presented a better learning performance at specific points.



(a) Test accuracy of IID data at SNR=10 dB (b) Test accuracy of non-IID data at SNR=20 dB

Fig. 4.6: Test accuracy under different scenarios

Wi-Fi transmission targets the SNR range of 10-30 dB [81] in a good transmission. When SNR=0 dB, there will be no valid packet due to transmission errors and packet retransmission. Our proposed method still achieves about 60% test accuracy when SNR=0 dB, as shown in Fig. 4.7. The proposed method is superior to the ECRT method at a very low SNR. Also, the high SNR gets better test accuracy than the low SNR.

To quantify the error impacts, the l_2-norm is applied. Fig. 4.8(a) shows the l_2-norm gradient error with 256-QAM on the MNIST dataset under non-IID conditions at SNR=10dB. The error accumulates linearly. The average error is used to evaluate the impact of the SNR

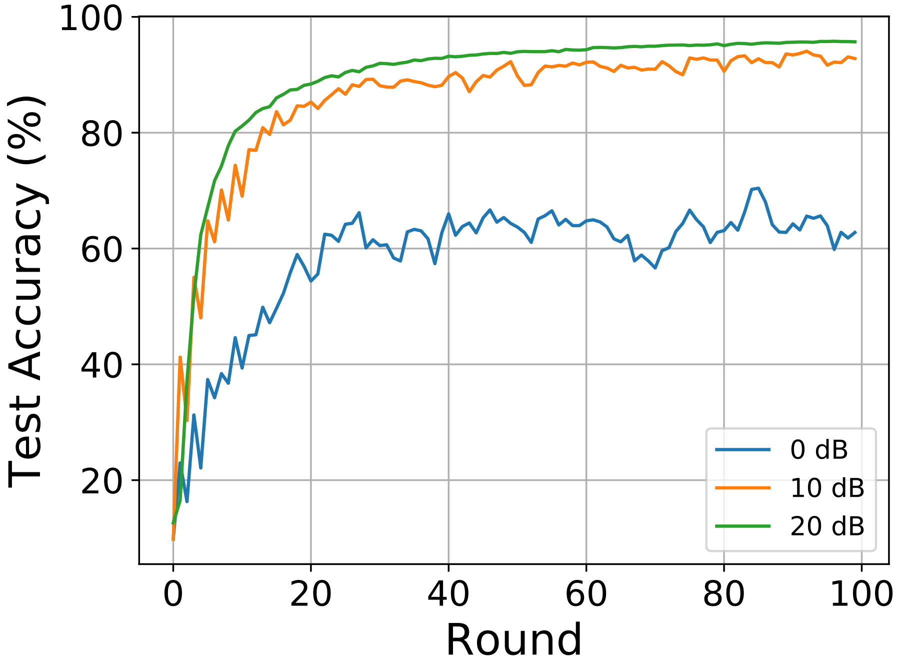
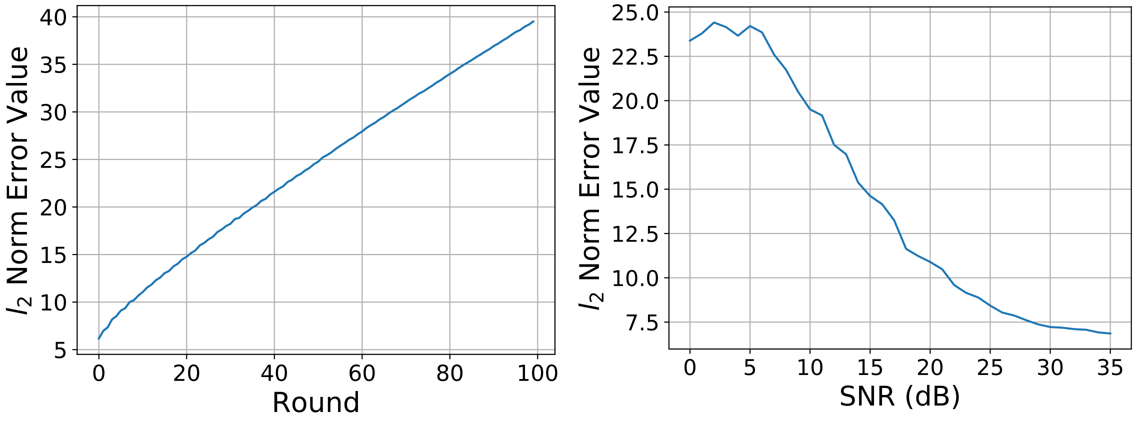


Fig. 4.7: Test accuracy of MNIST, non-IID with different SNR

on transmission errors, which is calculated in 100 rounds in each SNR. Fig. 4.8(b) presents the average error decrease with SNR increasing. This explains why learning performance is better in a high SNR regime.

The ECRT scheme ensures reliable gradient transmission through FEC and packet retransmission. On the contrary, native error transmission transmits the gradient without any protection, resulting in a flat learning curve, i.e., random guessing. Our proposed scheme leverages knowledge of gradient and gradient clipping. It outperforms naive error transmission, yielding significantly enhanced outcomes.

Communication time is compared in different transmission schemes. For the ECRT scheme, a practical IEEE 802.11 protocol with LDPC code is applied. The coding rate is set as 1/2. As reported in [82], for a code rate of 1/2 with a code length of 648, the minimum Hamming distance is 15. The error correction capability is 7 bits. The packet retransmission rate is 3.4% and 23% at SNR=20 dB and SNR=10 dB, respectively. The packet retransmission affects TCP throughput [83]. In Fig. 4.9, it takes 2× time for the ECRT scheme to achieve 80% test accuracy than the proposed scheme at SNR=20 dB.



(a) l_2 error norm at SNR =10 dB

(b) Average l_2 error norm at different SNR

Fig. 4.8: l_2 error norm

When SNR = 10 dB, it spends 3× time. Compared to the ECRT scheme, the proposed scheme saves time.

The gray coding in high-order modulation can protect the MSBs. The Fashion-MNIST dataset is used. In Fig. 4.10(a), the learning results with different modulations at the same SNR = 10 dB are presented. The learning results are similar. This underlines the resilience conferred by gray coding across varying modulation schemes.

Table 4.4: SNR for Target BER

BER	**QPSK**	**16-QAM**	**256-QAM**
4.36×10^{-2}	10 dB	16 dB	26.1 dB
4.91×10^{-3}	20 dB	26.1 dB	36.5 dB

Table 4.4 presents the SNR values for the target BER for different modulations. In Fig. 4.10(b), the learning performance with gray coding and 256-QAM surpasses both QPSK and 16-QAM when the BER is identical. The MSBs get better protection with 256-QAM modulation.

In FL, weighted averaging is used for model aggregation. The effects of the number of users are presented in Fig. 4.11. 20, 50, 80, or 100 users are selected in each FL round. In Fig. 4.11(a), the ECRT transmission is used, producing similar learning outcomes. In

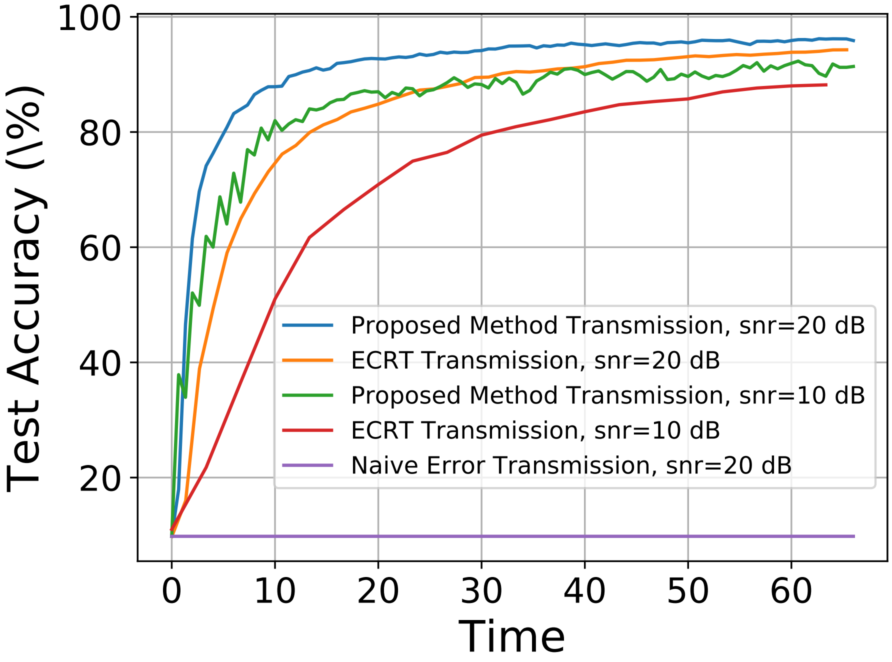


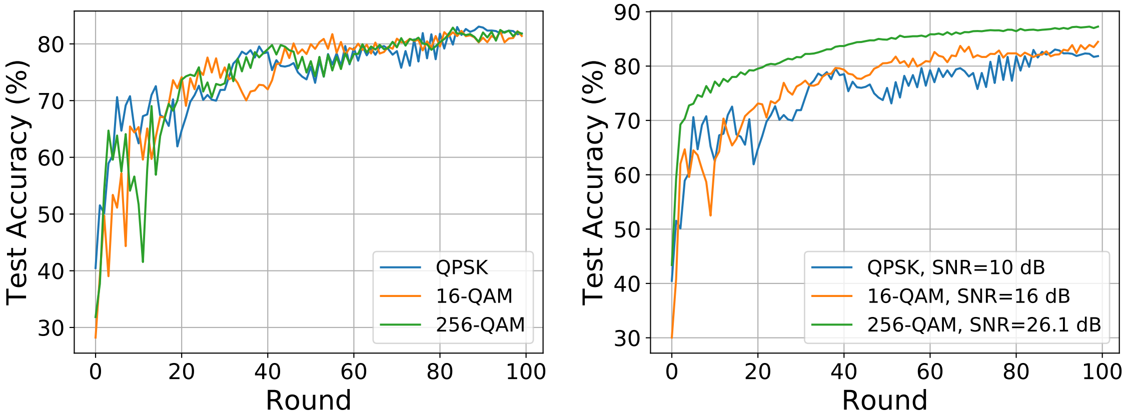
Fig. 4.9: Test accuracy of MNIST under non-IID

Fig. 4.11(b), it is simulated with QPSK modulation at SNR=20dB. When 20 users participate in the learning with transmission errors, the learning result experiences significant fluctuations. When 50, 80, or 100 users engage in learning, the test accuracy mirrors the ECRT transmission. The learning outcomes are enhanced by involving more users in the approximate communication.

In Fig. 4.12, the gradients are sparsified to 10%, 30%, 50%, 70% and 90%, respectively. 10% means that only 10% of the gradients are kept, and the best learning performance is achieved. Under approximate communication, the gradients with errors can also be destructive rather than constructive. From a time perspective, when the sparsification rate is 10%, while the indices of the kept gradients need to be transmitted in accurate channels, compression still saves close to 10× time.

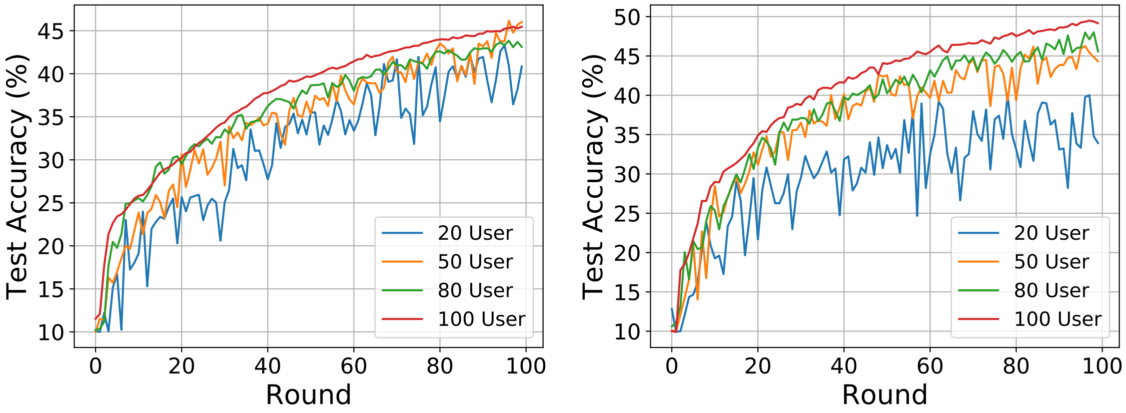
4.6 Chapter Conclusion

This chapter introduces a novel approach for transmitting the parameters of the FL model in wireless networks. Unlike the existing transmission methods, which rely on forward



(a) Test accuracy at the Same SNR=10 dB (b) Test accuracy at the Same BER $\approx 4.36 \times 10^{-2}$

Fig. 4.10: Test accuracy of Fashion-MNIST under IID



(a) Test accuracy using ECRT transmission (b) Test accuracy using our proposed method

Fig. 4.11: Test Accuracy of Cifar-10 under non-IID

error correction and packet retransmission to ensure reliable transmission, the proposed scheme involves gradient masking and approximate communication with transmission errors. Extensive simulations demonstrate the effectiveness of the proposed method, which is more time-efficient. A number of users participating in the learning process can reduce the effects of transmission errors. Gradual sparsification can further reduce communication costs and mitigate the side effects of approximate gradient transmission.

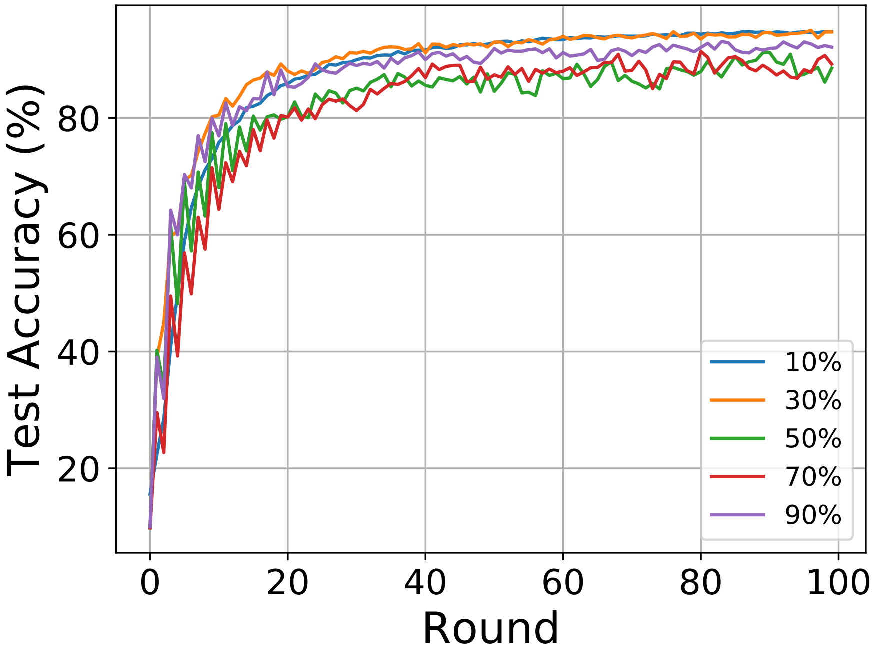


Fig. 4.12: Test accuracy of MNIST under non-IID with approximate communication and sparsification

CHAPTER 5

User Scheduling for Federated Learning Through Over-the-Air Computation

5.1 Introduction

In the last chapter, we applied approximate communication in FL model transmission. The gradient distribution and gradient clipping method to mitigate the gradient exploding problems motivated us to mask the received bits and constrain the received gradient values. The controlled error with the error resilience of the ML system makes the FL system obtain great learning performance with at least half of the time saved. This chapter explores different user scheduling policies in the transmission of FL models. Rather than using NOMA in Chapters 2 and 3, we use over-the-air computation (AirComp) as the transmission protocol in the uplink from clients to the server. It not only reduces communication time by allowing multiple clients to transmit the signal in the same channel simultaneously, as NOMA does, but also reduces computation time by aggregating the local model to build the global model in the air.

In [20], FL is presented through AirComp. AirComp uses the superposition nature of a wireless multiple-access (MAC) channel to compute multiple signals in the air. The FL system aggregates the received local models, constructs the global model, and distributes the global model to clients. The server only cares about the aggregated model rather than the individual local model. So AirComp is a good fit for FL model transmission. Since AirComp computes the global model in the air, the server cannot decode and get the individual local model. Therefore, it is computation-efficient and provides extra privacy protection for the clients.

The power control for AirComp in the fading channel is presented in [84] and [85]. The objective is to minimize the computation error caused by the analog transmission. The mean square error (MSE) of the received and transmitted signals is used to measure the computation error. Power control can significantly reduce MSE. In [39], the authors evaluated the FL learning performance under AirComp in both the digital and analog approaches. The analog approach learns faster and achieves better learning results than the digital one. Under the IID data distribution, the analog AirComp almost gets the learning performance as the error-free shared link. In [86], the FL learning rate in AirComp is optimized. The MSE and test accuracy are used to measure learning performance. However, no existing work has considered the user scheduling scheme for FL under AirComp, which can significantly affect learning performance.

This chapter considers FL via AirComp to reduce communication and computation time. The transmitter power control and receiver beamforming design are employed to minimize MSE. Three different user scheduling schemes are explored to study their effects on FL learning performance. One scheme considers the channel conditions from the communication perspective, another considers the significance of model parameters from the computation perspective, and the third is hybrid, which considers both the channel conditions and the significance of model parameters.

5.2 System Model

This chapter describes the AirComp communication protocol for FL. The FL system is considered to have a server and M clients. The server has N antennas, and each client has only one antenna. Under AirComp, multiple clients can transmit the model parameters simultaneously in the same channel, like in NOMA. Since channel conditions affect AirComp performance, poor client conditions can degrade MSE and FL learning performance. Implied by [20], a maximum number of clients can transmit via AirComp, which is assumed to be K here. Table 5.1 summarizes the main notation used.

The FL system is described in Chapter 1. Clients send local model parameters to the server for aggregation. The server performs the model aggregation to obtain the global model. With AirComp, computation and communication happen simultaneously in the air. The client waveforms are superposed to get the aggregated model.

Table 5.1: Summary of Notations

Notation	**Definition**
M; K; W	The total number of clients connected to the server; the maximum number of clients participating FL in each round; the intermediate number of clients when considering both the channel condition and the significance of the model.
N; T	The number of antennas in the server; the total number of communication rounds
$\mathbf{x}_k$; $\mathbf{y}_k$; $\boldsymbol{w}_k$;	Features of a data point sample on client k; corresponding label of data point; parameter set describe the mapping from $\mathbf{x}_k$ to $\mathbf{y}_k$
$F(\cdot)$; $f(\cdot)$; η	Global loss function; local loss function; learning rate
D_k; $\|D_k\|$	Dataset on user k; cardinality of the dataset D_k
$\boldsymbol{h_k}$; b_k; s_k	Channel vector of user k; transmitter scaling factor of user k; normalized local update at one time slot
P_0; $\phi_k(\cdot)$; $\psi(\cdot)$	Maximum transmit power; pre-processing function of user k; post-processing function at PS
$\boldsymbol{r}$; $\boldsymbol{a}$; $\boldsymbol{n}$	Received signal vector; receiver beamforming vector; additive noise
g; $\hat{g}$; τ	summation result before post-processing; estimation of g; normalization factor
S_K	Selected client set

The received signal at the server is given by

$$\boldsymbol{r} = \sum_{k=1}^{K} \boldsymbol{h}_k b_k s_k + \boldsymbol{n}, \tag{5.1}$$

where $||\boldsymbol{s}_k||_2^2 = I$, $\boldsymbol{n} \sim \mathcal{CN}(0, \sigma^2 \boldsymbol{I})$ is the noise vector. The transmit power constraint at client k is

$$E(|b_k s_k|^2) = |b_k|^2 \leq P_0, \tag{5.2}$$

where P_0 is the maximum transmit power.

The target function on the server side can be written as $v = \psi(\sum_{k=1}^{K} \phi(s_k))$, where $\phi_k(x) = |D_k|x$ is the pre-processing function on the client k, and $\psi(x) = \frac{1}{|D|}x$ is the post-processing function on the server side. Then, the summation of the transmitted signal is as follows.

$$g = \sum_{k=1}^{K} \phi_k(s_k). \tag{5.3}$$

Since beamforming is used on the receiver side, the value after beamforming is

$$\hat{g} = \frac{1}{\sqrt{\tau}}\boldsymbol{a}^{\mathsf{H}}\boldsymbol{r} = \frac{1}{\sqrt{\tau}}\boldsymbol{a}^{\mathsf{H}}\sum_{k=1}^{K}\boldsymbol{h}_k b_k s_k + \frac{\boldsymbol{a}^{\mathsf{H}}\boldsymbol{n}}{\sqrt{\tau}}. \tag{5.4}$$

MSE is used to measure the distortion of $\hat{g}$ with respect to g,

$$\begin{aligned}\mathsf{MSE}(\hat{g}, g) &= E(|\hat{g} - g|^2)\\ &= \sum_{k=1}^{K}|\frac{1}{\sqrt{\tau}}\boldsymbol{a}^{\mathsf{H}}\boldsymbol{h}_k b_k - \phi_k|^2 + \frac{\sigma^2||\boldsymbol{a}||^2}{\tau}.\end{aligned} \tag{5.5}$$

The transmitter power control parameter b_k and the receiver beamforming parameter $\boldsymbol{a}$ are optimized to minimize MSE. First, it is assumed that $\boldsymbol{a}$ is given and b_k can be optimized with a uniform forcing method such as [87]:

$$b_k = \sqrt{\tau}\phi_k\frac{(\boldsymbol{a}^{\mathsf{H}}\boldsymbol{h}_k)^{\mathsf{H}}}{||\boldsymbol{a}^{\mathsf{H}}\boldsymbol{h}_k||^2}. \tag{5.6}$$

Then, the normalization factor τ can be calculated as

$$\tau = P_0 \min_k \frac{||\boldsymbol{a}^{\mathsf{H}}\boldsymbol{h}_k||^2}{\phi_k^2}. \tag{5.7}$$

Now, the MSE problem (5.5) can be written as

$$\mathsf{MSE} = \frac{||\boldsymbol{a}^{\mathsf{H}}||^2\sigma^2}{\tau} = \frac{\sigma^2}{P_0}\max_k\frac{\phi_k^2||\boldsymbol{a}^{\mathsf{H}}||^2}{||\boldsymbol{a}^{\mathsf{H}}\boldsymbol{h}_k||^2}. \tag{5.8}$$

To get the best learning performance, the MSE needs to be minimized. The problem can be further written as

$$\min_{\boldsymbol{a}}\max_k\frac{\phi_k^2||\boldsymbol{a}^{\mathsf{H}}||^2}{||\boldsymbol{a}^{\mathsf{H}}\boldsymbol{h}_k||^2}. \tag{5.9}$$

This min-max problem can be reformulated as:

$$\begin{aligned} \min_{\boldsymbol{a}} \quad & ||\boldsymbol{a}||^2 \\ \text{s.t.} \quad & \frac{||\boldsymbol{a}^{\mathsf{H}}\boldsymbol{h}_k||^2}{\phi_k^2} \geq 1. \end{aligned} \tag{5.10}$$

Eq. (5.10) is a quadratically constrained quadratic programming problem (QCQP) with non-convex constraints, which is a hard NP problem [88]. In [87], semidefinite programming (SDP) is applied to solve the QCQP problem to find the initial solution. Then, successive convex approximation (SCA) is employed to improve the solution. The beamforming factor $\boldsymbol{a}$ can be solved this way, and all other parameters can be obtained. Minimum MSE can be achieved.

Algorithm 4 Beamforming Optimization by SDP and SCA

```
Initialize $\boldsymbol{a}$
SDP method to obtain $\boldsymbol{A}^*$
if rank($\boldsymbol{A}^*$) $\neq 1$ then
    $\tilde{\boldsymbol{a}}^* = \sqrt{\lambda_1}\boldsymbol{u_1}$
    Set $\boldsymbol{c_k} = [\Re(\tilde{\boldsymbol{a}}^{*\mathsf{H}}\boldsymbol{h}_k), \Im(\tilde{\boldsymbol{a}}^{*\mathsf{H}}\boldsymbol{h}_k)], \forall k$
    repeat
        SCA method to solve $\frac{||\boldsymbol{c_k}||^2}{\phi_k^2} \geq 1$ to obtain $\boldsymbol{a}$ and $\boldsymbol{c_k}$
    until criteria satisfied
else
    $\boldsymbol{a} = \sqrt{\lambda_1}\boldsymbol{u_1}$
end if
```

Algorithm 4 describes the SDP and SCA methods to optimize the receiver vector. Here, $\boldsymbol{A}^* = \min_{\boldsymbol{A}} \mathrm{Tr}(\boldsymbol{A})$, where $\mathrm{Tr}(\boldsymbol{A})$ is to obtain the trace of $\boldsymbol{A}$, $\boldsymbol{A} = \boldsymbol{a} * \boldsymbol{a}^H$. λ_1 is the largest eigenvalue of $\boldsymbol{A}^*$ and $\boldsymbol{u_1}$ is the corresponding eigenvector. $\boldsymbol{c_k}$ is the auxiliary variable. $\Re(\cdot)$ and $\Im(\cdot)$ are to get the real and imaginary parts of the complex value.

5.3 User Scheduling Policies

Three different user scheduling policies are considered. One is the channel condition-based scheduling. The clients with the highest channel gains will be selected. Another is the model-significance-based scheduling. Clients with the most significant model parameters will be selected. Hybrid scheduling first considers channel conditions to save energy, and only selected clients will begin local training. Then, the clients with the most significant model parameters will be scheduled to upload their local models.

5.3.1 Channel Condition Based Scheduling

The channel condition-based scheduling selects the K users with the highest channel gains, i.e.,

$$S_K = \max_{[K]}\{||\boldsymbol{h}_1(t)||_2, \dots, ||\boldsymbol{h}_M(t)||_2\}, \tag{5.11}$$

where, $||\boldsymbol{h}_k(t)||_2 = \sqrt{\sum_{i=1}^{N} |h_k^i(t)|^2}$ is the l_2-norm channel gain of client k. Each client needs to send a small amount of information to the server for channel estimation. Compared with the transmission of the model parameters, the time for channel estimation can be ignored.

From equation 5.8, when the channel gain $\boldsymbol{h}_k$ is the only variable, other parameters are fixed. The higher the channel gain, the smaller the MSE value. Channel condition-based scheduling selects clients with the highest channel gains to upload their local model parameters. Unselected clients can skip local training in this round and save computational energy.

5.3.2 Model Significance Based Scheduling

In model significance-based scheduling, model significance is used as a criterion to select clients. The l_2-norm of the model parameter is applied to evaluate the significance of the model. Each client first performs local training and obtains the local model parameters $\boldsymbol{w}_k^t$. And then sends its l_2-norm $||\boldsymbol{w}_k^t||_2$ to the server. The server selects the clients with the largest $||\boldsymbol{w}_k^t||_2$ values,

$$S_K = \max_{[K]}\{||\boldsymbol{w}_1(t)||, \dots, ||\boldsymbol{w}_M(t)||\}. \tag{5.12}$$

This scheme requires all clients to execute local training and send their l_2-norm of the model parameter to the server. This causes extra energy waste for the clients that will not be selected later. Additionally, stragglers in the FL system can take a long time to complete local training, increasing the overall learning time of the FL.

5.3.3 Hybrid Scheduling

The channel conditions and the significance of the model can both affect the FL learning performance. Therefore, both are considered in the hybrid scheduling. The server selects W clients according to channel conditions as in channel condition-based scheduling. Then, K clients with the highest model significance are selected from the selected W clients. Here, $K \leq W \leq M$. In this way, only W clients need to do local training to get the local model.

Channel-based scheduling reduces the extra energy costs for unselected clients, while model significance-based scheduling helps directly improve FL learning performance. Hybrid scheduling takes advantage of both.

5.3.4 Time Complexity Analysis

The end-to-end time to complete learning performance is critical for latency-constrained applications. We analyze the communication and computation time to complete one FL round. For simplicity, the computational capacity for each client is assumed to be identical. The local computation time to complete the ML task is t_p. The communication time to transmit a small amount of information to the server for channel estimation is t_o, and the communication time to upload model parameters is t_u. Table 5.2 summarizes the corresponding time complexity.

Table 5.2: Complexity Analysis

	Channel Based Scheduling	Model update Based Scheduling	Hybrid Scheduling
Communication Time	$M * t_o + K * t_u$	$K * (t_o + t_u)$	$M * t_o + K * t_u$
Computation Time	$K * t_p$	$M * t_p$	$W * t_p$

5.4 Simulation

This section presents the FL simulation under AirComp with different user scheduling schemes. Learning performance is evaluated using test accuracy.

5.4.1 Simulation Settings

The channel parameters are given as follows. The total number of clients in the FL system is $M = 1000$. Clients are uniformly distributed in a disk area with a radius of 500 m. The transmit signal to noise ratio $\frac{P_0}{\sigma^2}$ is fixed at 42 dB, and the channel path loss exponent is $\alpha = 3$. The channel is assumed to be constant during one FL round but varies across rounds. The number of antennas on the server is $N = 4$. The number of clients who can participate in AirComp uplink in each round is $K = 10$. And the intermediate number of clients scheduling in hybrid scheduling is $W = 20$.

MNIST dataset is applied for ML tasks with LeNet-300-100 neural networks. Hyperparameters are summarized in Table 5.3. The data are distributed non-IID to make the proposed scheme more convincing.

Table 5.3: Hyperparameters

Learning rate size (η)	**Batch size ($\mathcal{B}$)**	**FL Round (T)**	**Training set size**	**Testing set size**
0.01	10	60	90%	10%

5.4.2 Simulation Results

Fig. 5.1 presents the FL test accuracy under channel condition-based scheduling and random channel scheduling. Random channel scheduling randomly selects clients with different channel conditions uniformly. Channel condition-based scheduling achieves higher test precision during the learning process than random channel scheduling. However, it experiences more significant fluctuations. The MSE obtained in the channel condition-based scheduling can be much smaller than that obtained by random scheduling: the non-IID

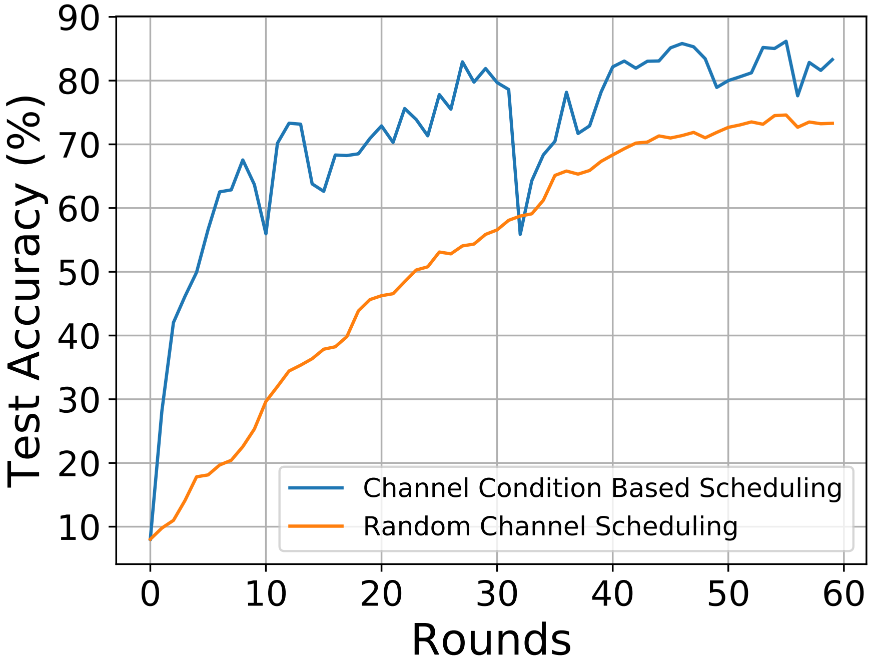


Fig. 5.1: Test accuracy under channel condition-based scheduling

data distribution results in large fluctuations. For random channel scheduling, learning performance is dominated by channel conditions rather than data distribution. Therefore, it experiences less fluctuation.

In Fig. 5.2, the test accuracy of the model significance-based and random model scheduling is given. Model significance-based scheduling is smoother than random model scheduling. The difference between model significance-based scheduling and random model scheduling is smaller than channel-based scheduling.

Fig. 5.3 shows three different user scheduling schemes. Hybrid scheduling achieves the learning performance between channel-based scheduling and model significance-based scheduling. Hybrid scheduling balances learning performance and energy costs.

5.5 Chapter Conclusion

AirComp can further reduce the communication and computation time needed to transmit the FL model. Three different user scheduling schemes are used to investigate the effects of user scheduling on FL performance. The model significance-based scheduling directly af-

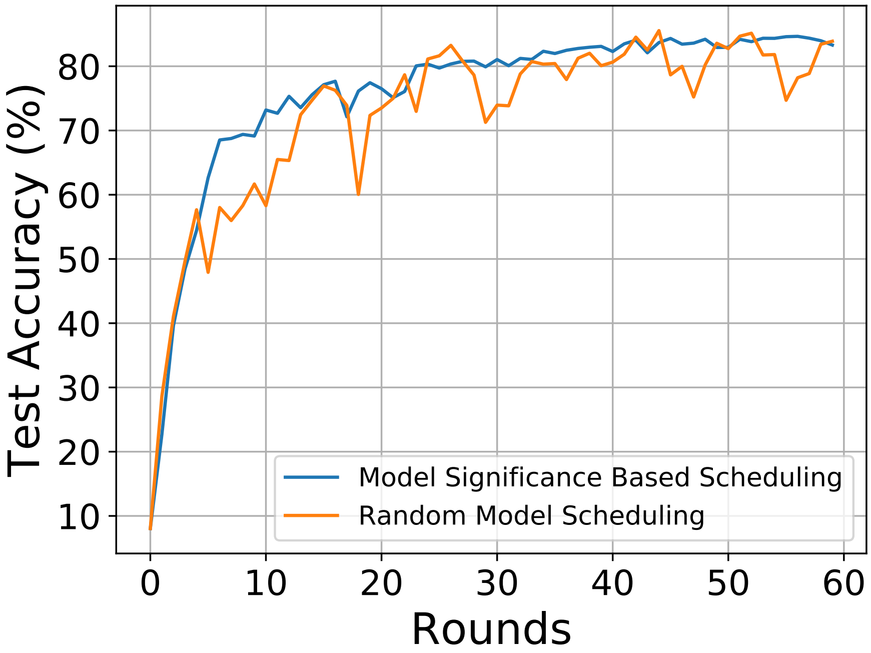


Fig. 5.2: Test accuracy under model significance based scheduling

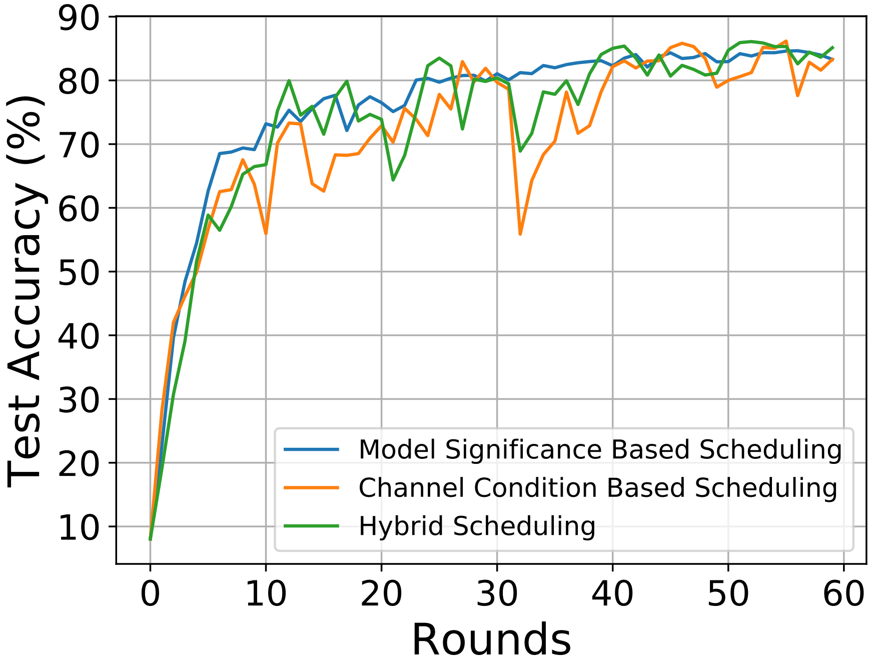


Fig. 5.3: Test accuracy under three different scheduling

fects learning performance and achieves the best test accuracy. Channel condition-based scheduling saves energy while experiencing higher learning fluctuations. Taking into account both the channel condition and the significance of the model, hybrid scheduling achieves a good trade-off.

CHAPTER 6

A New Implementation of Federated Learning for Privacy and Security Enhancement

6.1 Introduction

In previous chapters, we studied the method to reduce communication time and speed up the learning process in FL. Different communication protocols have been used, including NOMA, approximate communication, and AirComp. Adaptive gradient compression is used to satisfy channel capacities. User scheduling and power allocation schemes are also considered to improve FL learning performance. AirComp can also provide better privacy protection by hiding the individual local model from the server and eavesdroppers.

This chapter will focus on security and privacy issues in FL. FL is designed to provide privacy protection for client data by sharing models rather than sharing raw data. However, several problems have occurred that make the FL system insecure. First, FL is a distributed system with multiple clients. Some clients may be attacked and controlled by malicious parties. Malicious clients can launch Byzantine attacks [89] to send poisoning models to the server and degrade learning performance. Furthermore, data privacy can still be leaked even with model sharing. Membership inference attacks [90] can infer the data from the gradients and the model information. These problems challenge the security and privacy of FL, especially in wireless networks.

Byzantine attacks refer to attacks initiated by internal nodes of the system. In FL, malicious clients send the poisoning model parameters to the server. The attack can be a data poisoning attack or a model poisoning attack. In data poisoning attacks, the labels are changed, or false data are injected to produce falsified model parameters and mislead the global model. In model poisoning attacks, malicious clients manipulate local models and directly affect the global model. It is challenging for the server to examine the local model parameters of each client and identify the poisoning model. However, malicious clients may occasionally send poisoning information to the server to reduce the probability of identification. When a secure aggregation protocol is applied, and the individual local client model is invisible to the server, the server has no access to local models and cannot distinguish the malicious ones.

Various works have been performed to defend against Byzantine attacks in distributed ML. In [91], the researcher applied the variational autoencoder (VAE) to extract the feature of the local models and identify malicious clients with a dynamic threshold. However, it can be challenging to determine the threshold. In other words, they aim to mitigate the adverse effects rather than identify malicious clients. Krum aggregation [92] selects some clients as benign and trusted clients and estimates the true center based on the majority and the squared distance. This may cause the learned global model to be biased, especially when the data are non-IID distributed. Geometric median aggregation [93] considers compounding to mitigate the adverse effects. However, it requires knowing the individual client model, which may arouse membership inference attacks.

Membership inference attacks can cause disclosure of privacy. The attacker can infer the data with only gradient and model information. A classical method to improve privacy protection is differential privacy (DP) [94]. It typically adds random noise to the information and hides the real information. In FL, the local model can be uploaded with random noise added, and the real local model is hidden. Since global aggregation uses the weighted average, the noise effects can be ignored when the number of clients is large enough. Another method to protect information privacy is to hide the individual model from the eavesdroppers, i.e., only the aggregated model is accessible. The server only cares about the aggregated model, so hiding the individual model does not affect the learning process. Secure aggregation (SecAgg) [95] applies vector masking to hide the individual model from the server. AirComp [96] can aggregate the local model in the air and produce the aggregated model for the server. These methods can protect user privacy from eavesdroppers and dishonest servers.

This chapter will introduce a novel model update-based (MUB) aggregation method to defend against Byzantine attacks and improve FL security. It will be proven to be resilient to additive noise attacks and sign-flipping attacks. The individual client model initialization (ICMI) scheme initializes the global model of each client rather than accepting a global model from the server. By combining the MUB and ICMI schemes, FL's security and privacy protection are enhanced.

6.2 System Model

6.2.1 Classical FL

As mentioned in Chapter 1, the FL system considers M clients in total. K clients can be scheduled each round to upload their local model parameters. When the gradient information is uploaded, it is called FedSGD. For FedAvg, the local model information is transmitted to the server. For FedAvg, the local SGD can be written as:

$$w_t^k = w_t - \eta \nabla F_k(w_t). \tag{6.1}$$

The global aggregation at the server is

$$w_{t+1} = \sum_{k=1}^{K} \frac{|D_k|}{|D|} w_t^k. \tag{6.2}$$

In Fig. 6.1, each FL round is divided into two parts: global aggregation and local training. The server initializes the global model and sends it to clients in the first round.

6.2.2 Model Update

The global aggregation in Equation (6.2) applies a smile arithmetic averaging algorithm. It is highly efficient. However, it was not robust enough to deal with Byzantine attacks. The model update is defined as the difference between the current and previous models. For the local model update

$$u_t^k = w_t^k - w_t^{k'}, \tag{6.3}$$

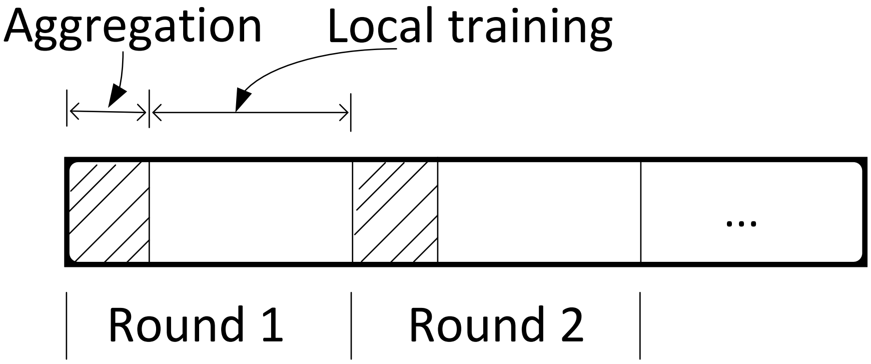


Fig. 6.1: FedAvg FL round

where w_t^k is the local model after local training, and $w_t^{k'}$ is the local model before local training. The global model update is then defined as

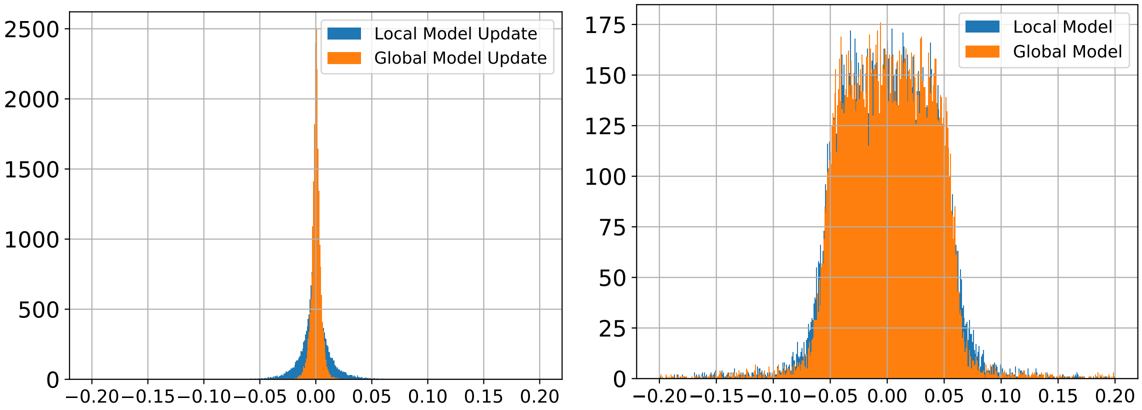
$$u_{t+1} = \sum_{k=1}^{K} \frac{|D_k|}{|D|} u_t^k. \tag{6.4}$$

The global model update rather than the global model is sent back to the clients from the server. So, the local model before local training $w_t^{k'}$ is not identical to the global model. In addition, local models before local training vary between clients. The local model before local training $w_t^{k'}$ is calculated based on the local model in the last round w_{t-1}^k and the most recent global model update u_t as:

$$w_t^{k'} = w_{t-1}^k + u_t. \tag{6.5}$$

The distribution of the difference between the model and the update of the model is given in 6.2 learning in a non-IDD data setting. Compared to the model, the distribution of the model update is more focused. In Fig. 6.2(a), the distribution of local model update u_t^k is close to the distribution of global model update u_t. Similarly, the distribution of the local model w_t^k in Fig. 6.2(b) is close to the distribution of the global model w_t.

In [97], the authors investigated the model distribution. They proposed the norm-clipping approach to make the norm of the model parameters small enough so that the



(a) Distribution of model update

(b) Distribution of model

Fig. 6.2: Distribution of model update and model with non-IID data in FL

server is less susceptible to the poisoning models of backdoor attacks. The norm clipping can effectively defend against backdoor attacks with little interference from the primary task. Based on the comparison of the distribution of model update and model, model update can be a natural substitute for the norm-clipped model. It should work well to defend against Byzantine attacks.

6.2.3 Initial Client Model Initialization

In classical FedAvg, the server initializes the global model and then sends it to clients. Thus, each client has the same starting point for learning. The initial client model initialization (ICMI) scheme initializes the model on clients rather than on the server. This helps to hide the initial model of the clients from the server and eavesdroppers. Since both model initialization methods randomly initialize the model, the ICMI scheme does not affect the convergence of learning.

6.3 Proposed Method

This section will first describe the model update-based (MUB) FL for defending against Byzantine attacks. Then, the ICMI scheme with secure aggregation hides individual client models to prevent membership-inference attacks. Finally, MUB and ICMI are combined to improve FL security and privacy.

6.3.1 MUB FL

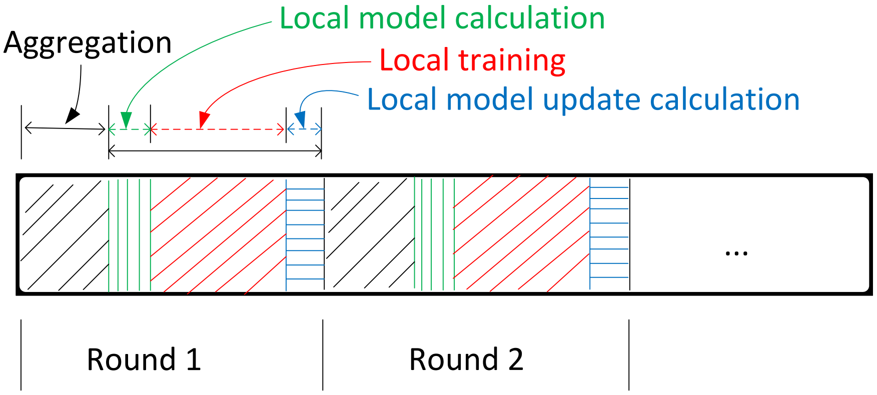


Fig. 6.3: MUB FL round

In classical FedAvg, there are two stages in each FL round: 1) global model aggregation at the server and 2) local model training at the clients. For the MUB FL, as shown in Fig. 6.3, the aggregation stage aggregates the model update rather than the model, and three sub-stages are followed, that is, local model calculation, local training, and local model update calculation. The three substages can be summarized as follows:

$$w_t^{k'} = w_{t-1}^k + u_t, \tag{6.6a}$$

$$w_t^k = w_t^{k'} - \eta \nabla F_k(w_t^{k'}), \tag{6.6b}$$

$$u_t^k = w_t^k - w_t^{k'}. \tag{6.6c}$$

In ML, the gradient is usually defined as $g = \nabla F(w)$. Then, the local gradient in FL can be written as $g_t^k = \nabla F_k(w_t)$. According to equations (6.6b) and (6.6c), we can rewrite $u_t^k = -\eta g_t^{k'}$. This only applies to one local iteration performed in one FL round. However, multiple local iterations are executed in FL to reduce the communication frequency and save bandwidth. Then equation (6.6b) becomes

$$w_t^k(0) = w_t^{k'}, \tag{6.7a}$$

$$w_t^k(j+1) = w_t^k(j) - \eta \nabla F_k(w_t^k(j)). \tag{6.7b}$$

$w_t^{k'}$ serves as the local learning starting point as in equation (6.7a). Then, multiple local iterations are executed in Equation (6.7b).

Classical FedAvg sends the global model to clients after global aggregation. The learning starting point for each client is identical in each round. For MUB FL, the local model in the first round is $w_1^k = w_1^{k'} - \eta \nabla F_k(w_1^{k'})$, where $w_1^{k'} = w_0^k + u_1$. When $w_0^k = w_1$ and $u_k = 0$, it is identical to the classical FedAvg. However, starting from the second round, the learning starting point for each client in MUB FL becomes different. In MUB FL, the global model update u_t is sent back to the clients after aggregation. Therefore, the global model update is identical for all clients. However, the local model w_{t-1}^k differs across clients. According to Equation (6.6a), the local learning starting point $w_t^{k'}$ is diverse. This is the main difference between the classical FedAvg and the MUB FL. Although the local learning starting point in MUB FL differs between clients, MUB FL can still achieve convergence similar to classical FedAvg without any attacks.

6.3.2 ICMI FL

As mentioned above, the ICMI scheme hides the initial model to protect user privacy. However, when ICMI is applied with classical FedAvg, the local client model in the following round is still available to eavesdroppers. Membership inference attacks can still be executed. Extra operations are needed to hide the individual client models. Secure aggregation (SecAgg) or AirComp can securely aggregate local models. Only the aggregated model is available to the server and eavesdroppers. Since the local model is hidden, eavesdroppers cannot infer the local data.

6.3.3 MUB-ICMI FL

Rather than combine ICMI with SecAgg or AirComp, ICMI can be combined with the MUB scheme. In MUB FL, the model update is transmitted rather than the model. With ICMI, local models in each round are hidden from the server and eavesdroppers. The MUB-ICMI scheme can improve both security and privacy. The MUB-ICMI algorithm is summarized in Algorithm 5.

Algorithm 5 MUB-ICMI FedAvg

```
Each client initializes $w_0^k$, server initializes $u_1 = 0$
**Server executes:**
  **for** each round t=1,2,... **do**
    $K \leftarrow \max(C \cdot M, 1)$
    $S_t \leftarrow$ (random set of $K$ clients)
    **for** each client $k \in S_t$ **in parallel do**
      $u_t^k \leftarrow$ ClientUpdate($k$, $u_t$)
    **end for**
    $u_{t+1} \leftarrow \sum_{k=1}^{K} \frac{|D_k|}{|D|} u_t^k$
  **end for**
**ClientUpdate($k$, $u_t$)**: // *Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  $w_t^{k'} \leftarrow w_{t-1}^k + u_t$
  $w_t^k(0) \leftarrow w_t^{k'}$
  **for** each local iteration $j$ from 0 to $N-1$ **do**
    **for** batch $b \in \mathcal{B}$ **do**
      $w_t^k(j+1) \leftarrow w_t^{k'}(j) - \eta \nabla F_k(w_t^{k'})(j)$
    **end for**
  **end for**
  $u_t^k \leftarrow w_t^k(N) - w_t^{k'}$
  return $u_t^k$ to server
```

6.4 Simulation

This section introduces the simulation settings and results. We first show the convergence of the MUB FL, ICMI FL, and MUB-ICMI FL algorithms and the classical FedAvg algorithm in image classification tasks without attacks. Then, the four schemes are tested under Byzantine attacks, i.e., additive noise attacks and sign-flipping attacks. The MUB-ICMI FL still achieves a good learning performance in defending against Byzantine attacks.

6.4.1 Simulation Settings

The FL system here is considered to have $M = 100$. For simplicity, we allow all clients to participate in the learning process in each round. The MNIST image dataset is used for image classification tasks. Two machine learning models, multilayer perception (MLP) and convolutional neural networks (CNN), are applied. For MLP, only one hidden layer is used. For the CNN model, two convolutional layers are followed by the max-pooling layers and two fully connected layers at the end. The IID and non-IID data distributions will be explored to verify the effectiveness of the proposed scheme with statistical heterogeneity. The test runs on the entire test dataset with global models. In MUB FL, the global model is calculated by accumulating the global model update from the initial training. The hyperparameters of the learning model are the learning rate $\eta = 0.01$, the batch size $B = 5$, and the local iteration $N = 2$.

6.4.2 Simulation Results

First, the convergence of classical FedAvg, MUB FL, ICMI FL, and MUB-ICMI FL can be demonstrated using MLP and CNN models in IID and non-IID data distributions without any attacks. In Fig. 6.4, we show the test accuracy of the MNIST non-IID dataset with the CNN model without any attacks. The four schemes achieve similar convergence after 200 training rounds. The MUB scheme achieves slightly better learning performance at the initial learning stage. This may be because the non-IID data cause less impact on MUB FL than classical FedAvg. The MUB-ICMI FL is slightly worse due to the double effects of ICMI and MUB. Similar learning performance is achieved when the data is IID or the model is MLP. All four schemes converge.

Additive noise and sign-flipping attacks are used here to verify the effectiveness of the proposed algorithm against Byzantine attacks. Additive noise attacks add random Gaussian noise to the transmitted information before sending it to the server. Malicious clients want

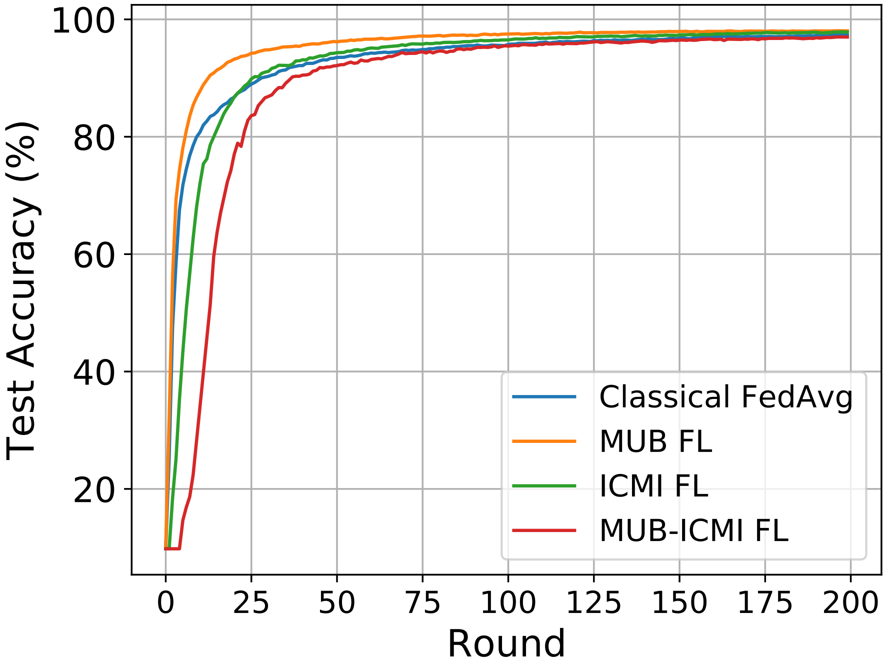
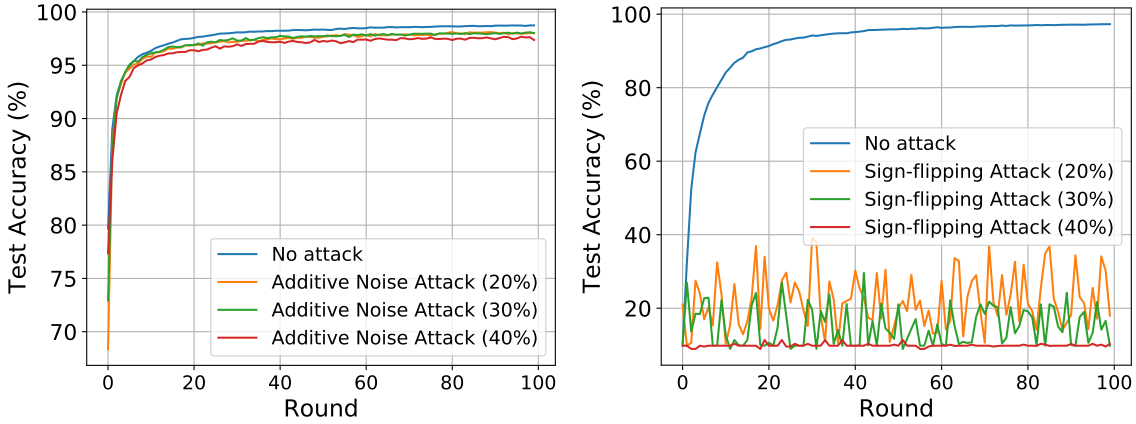


Fig. 6.4: Test accuracy of non-IID data with CNN model without any attacks

to affect the global model with random noise and large power. However, it can be easily detected by calculating the l_2-norm. So, the random noise power should align with the true information. The sign-flipping attacks flip the sign of each parameter in the information transmitted, while keeping the magnitude unchanged. It is difficult to detect and, thus, more harmful. The number of malicious clients is assumed to be 0, 20%, 30%, and 40% of the total of clients.

First, we show the classical FedAvg learning performance under these attacks. In Fig. 6.5, the test accuracy of IID data with additive noise attacks using classical FedAvg is shown in 6.5(a). The test accuracy of non-IID data with sign-flipping attacks is given in 6.5(b). Learning performance worsens with more clients becoming malicious and sending falsified information to the server. With 40% malicious clients, FL learns nothing under sign-flipping attacks in non-IID data distributions. Compared to additive noise attacks, sign-flip attacks can cause worse effects on FL learning. In the following, we will only show the test result of non-IID data distribution under sign-flipping attacks.

Fig. 6.6 presents the test accuracy of non-IID data with the MLP model using the



(a) IID data with additive noise attacks

(b) Non-IID data with sign-flipping attacksl

Fig. 6.5: Test accuracy with CNN model using classical FedAvg

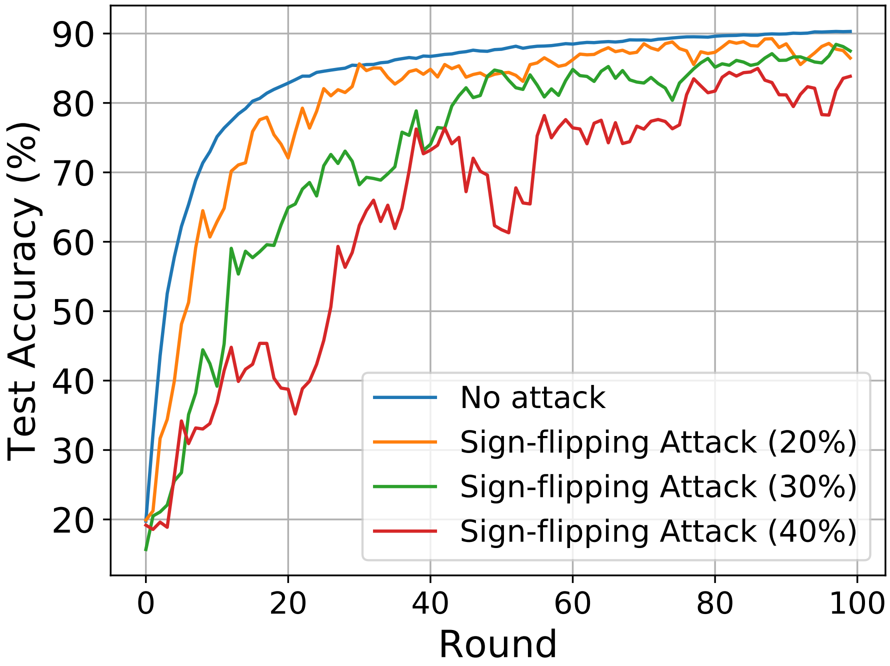


Fig. 6.6: Test accuracy of non-IID data with MLP model using MUB scheme

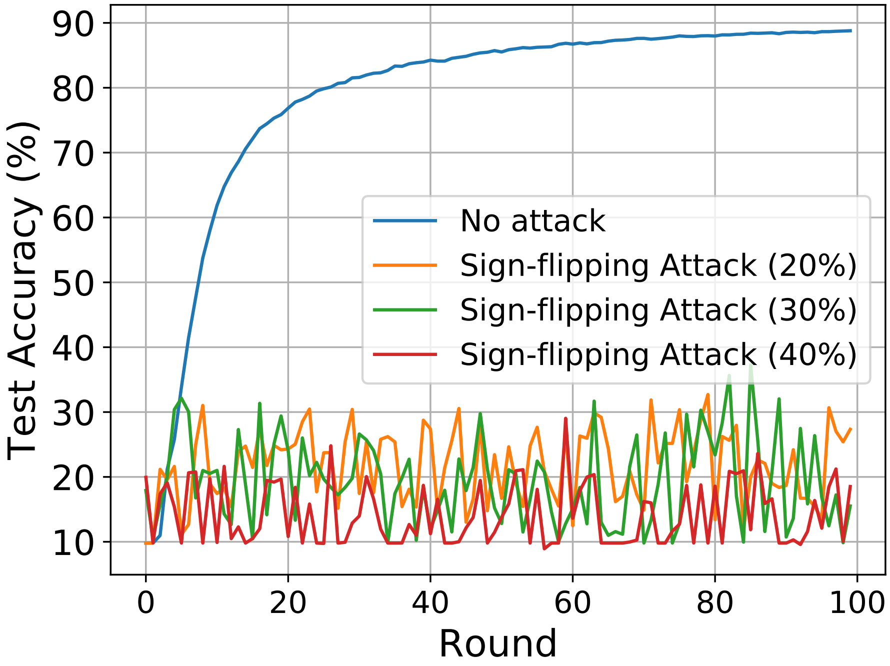


Fig. 6.7: Test accuracy of non-IID data with MLP model using ICMI scheme

MUB scheme. Learning experiences large fluctuations. However, it achieves good final test accuracy even in the worst case with 40% malicious clients after 100 learning rounds. The ICMI scheme achieves learning performances similar to those of the classical FedAvg, as shown in Fig. 6.7. It is designed to enhance privacy protection rather than defend against Byzantine attacks.

Finally, in Fig. 6.8, the test accuracy of the MUB-ICMI scheme under sign-flipping attacks is given. It is trained with 200 Fl rounds. For scenarios with 20% and 30% malicious clients, the final test accuracy is comparable to the scenario with “No attack”. Even in the worst case with 40% malicious clients, the learning result is much better than classical FedAvg.

6.5 Chapter Conclusion

In this section, we focus on security and privacy issues in FL. The model update-based (MUB) FL is proposed to defend against Byzantine attacks. An individual client model initialization (ICMI) scheme is introduced to hide the initial model and enhance privacy.

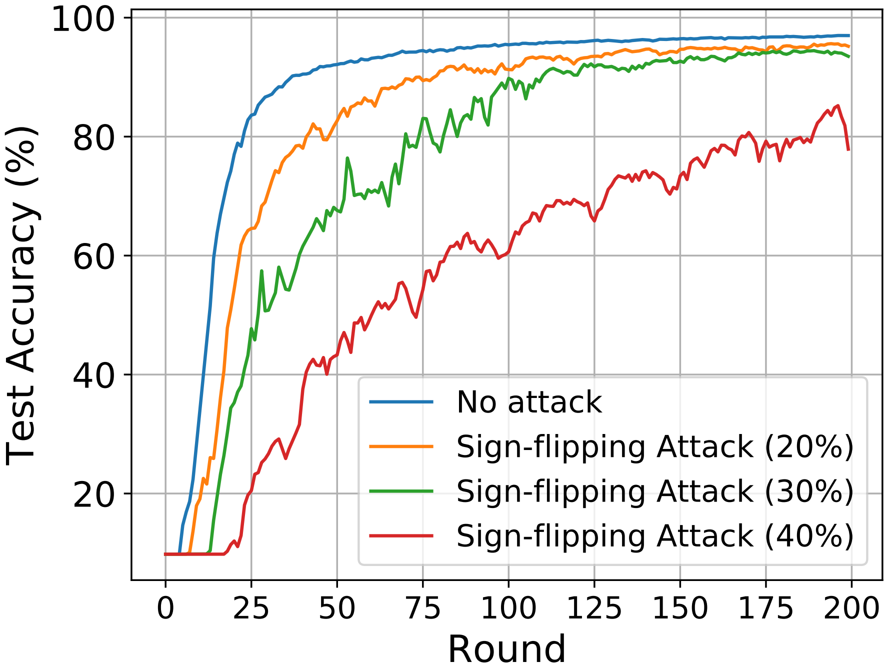


Fig. 6.8: Test accuracy of non-IID data with CNN model using MUB-ICMI scheme

The combined MUB-ICMI scheme can effectively improve security and privacy protection in FL. Tremendous simulations have been performed in different scenarios to prove the effectiveness of the proposed schemes.

CHAPTER 7
CSMAAFL: Client Scheduling and Model Aggregation in Asynchronous Federated Learning

7.1 Introduction

In the previous chapter, we studied the method for improving the security and privacy of the FL system. Model update-based (MUB) aggregation can improve the system's resilience against Byzantine attacks. The individual client model initialization (ICMI) scheme can be combined to hide the individual local model and enhance the privacy of local data.

This chapter introduces asynchronous FL to solve heterogeneous problems in the FL system. The FL system is a distributed ML system with many clients performing real training. And the server only aggregates the locally trained models and sends the global model back. The server has no control over the client's participation in the learning process. The server waits for a specific time duration or for a specific number of clients to upload their model parameters, then performs the aggregation. The clients in the FL system are equipped with various computing, memory, and storage capabilities. Computing-constrained clients may experience delays and become stragglers when processing large volumes of data. This would significantly affect learning performance or speed during the learning process.

In [1], the researchers proposed to select a subset of clients in each round, and it is not necessary to wait for all clients to complete their local training. However, straggler issues are not fully addressed as user scheduling schemes are not specified. When the aggregation strategy is to select a fixed number of clients, the waiting time is not yet determined. A predefined synchronous window was proposed in [98] to aggregate as many clients as possible in a given time. However, slow clients may not be able to contribute to the global model. In [99], an adaptive computation scheme was proposed in a resource-constrained system where clients can perform local computations based on local computational capabilities. The server still needs to wait for the clients to upload their locally trained models.

Asynchronous FL (AFL) [100] can be applied to deal with straggler problems. The local model computation at the clients and global aggregation at the server can be decoupled. The local computation becomes a non-blocking step for the global aggregation. In [25], an online AFL algorithm with non-IID data is proposed. The server starts the aggregation once it receives the local model parameter from a single client. There is no waiting time for other clients to complete their local computations and upload the model. The newly aggregated model is then returned to the clients who consider it "ready". However, the "ready" clients are not defined. And there is no user scheduling to select which client can upload its local model. Similarly, in [101], scheduling and aggregation are decoupled, but the selection criteria for user scheduling are not defined. AFL over wireless networks is considered in [102], where the aggregated global model is broadcast to all clients. Clients need to determine whether to continue local training using the previous global model or stop local training and apply the latest global model. This can cause significant energy and time waste for the clients who decide to continue their local computation with previous global models. The AFL may cause model stale problems due to the training being asynchronous. Some clients may start the local training with the latest global model, while others may train with an old model. The staleness is measured using the distance between the current and stale global models. In [103], a Euclidean distance-based adaptive federated aggregation method was presented to solve the model stale problems in AFL. This requires the server to store all global models, resulting in high storage consumption.

This chapter introduces a new AFL framework with client scheduling and model aggregation. Unlike the existing work, the aggregated model is exclusively returned to the client, who just uploads the local model. This eliminates the need to select the "ready" clients or broadcast them to all clients. We introduced a client scheduling scheme to consider computational capabilities and client fairness. A model aggregation algorithm is developed to address the model staleness problem in AFL. The global iteration difference is used here as the metric for staleness. And only one hyperparameter is introduced to include time and other factors. Also, a detailed analysis of the time complexity of AFL and synchronous FL (SFL) is given.

7.2 System Model

The FL system in previous chapters is a synchronous communication system in which the server performs aggregation after receiving local models from a fixed number of clients or after a fixed time has elapsed. In AFL, the server aggregates once the local model is received from a single client. This allows the server to be updated continuously without waiting.

7.2.1 Synchronous Federated Learning

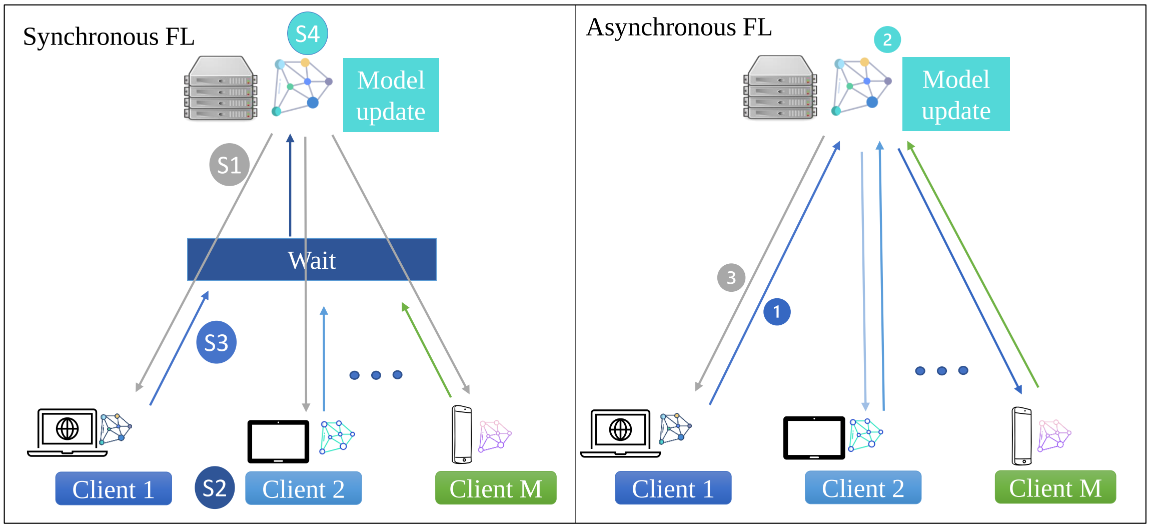


Fig. 7.1: Synchronous vs asynchronous FL

In 7.1, the system model for SFL (left) and AFL (right) is given. In SFL, there are four steps, as mentioned in Chapter 1. In step (S4), the aggregation of local models to obtain the global model will wait for multiple clients to complete the local training and uploading of the model in Step (S3). There is a "wait" stage in SFL to block global aggregation until the conditions are met.

The server initializes the global model as w_0. The clients perform local training as follows:

$$w_t^m = w_t - \eta \nabla F_m(w_t), t = 0, 1, 2, ..., \tag{7.1}$$

where w_m^t is the local model of client m at round t after local training, w_t is the global model and learning starting point for all clients at round t. This occurs in step (S2).

The global aggregation is defined as follows:

$$w_{t+1} = \sum_{m=1}^{M} \alpha_m w_t^m, \tag{7.2}$$

where we assume the server received local models from M clients during the waiting time, and α_m is the aggregation coefficient of client m. α_m is usually calculated as $\alpha_m = \frac{|D_m|}{\sum_c |D_c|}$, where $|D_m|$ is the data size at client m, and $\sum_{m=1}^{M} \alpha_m = 1$.

7.2.2 Asynchronous Federated Learning

In previous works, the AFL aggregates the local model once the server receives it from a single client. However, the aggregation detail varies in different implementations. Here, we use the global aggregation number to track the learning process. The term “iteration” represents the global aggregation number to distinguish the “round” in SFL. The aggregation model is defined as follows:

$$w_{j+1} = \beta_j w_j + (1 - \beta_j) w_i^m, \tag{7.3}$$

where i and j are global iterations in AFL. w_j and w_{j+1} represents the global model in iterations j and $j + 1$ on the server. w_i^m is the local model of the client m after local training using the global model at iteration i. The client m may use a new or an old global model to train the local model. It appears in Equation (7.3), meaning it was selected in iteration j. $\beta_j \in (0, 1)$ is the aggregation coefficient in AFL. While uploading the local model to the server by the client m, other clients can perform local training or wait for the channel to idle. After global aggregation, the updated global model w_{j+1} will return to the client m. This is a complete global iteration in AFL, including 1) local training, 2) global aggregation, and 3) receiving an updated global model. Only one client participates in each global iteration. The client m will continue to conduct local training as follows:

$$w_{j+1}^{m} = w_{j+1} - \eta \nabla F_m(w_{j+1}). \tag{7.4}$$

AFL allows the server to always perform the aggregation rather than waiting for a fixed number of clients to complete the local training and model uploading.

7.2.3 Time Comparison

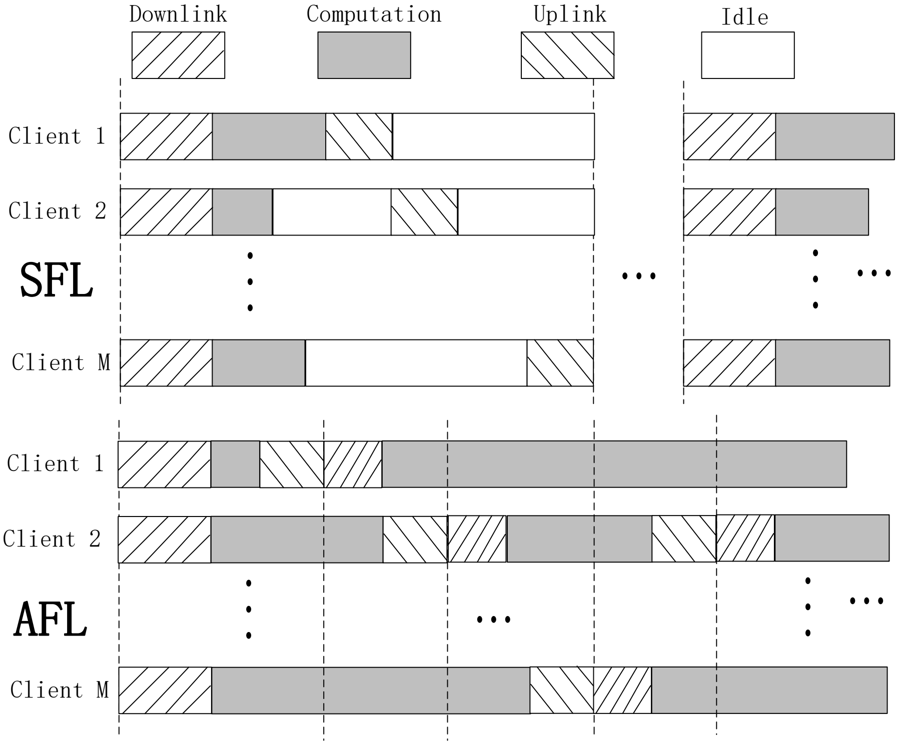


Fig. 7.2: Time comparison between SFL and AFL

In Fig. 7.2, the learning procedure of SFL (top) and AFL (bottom) is given. There is idle time for some clients in SFL while clients continue learning in AFL. Clients in

AFL take full advantage of the time to perform local training. To understand the AFL quantitatively, the performance of SFL and AFL is compared from the time perspective. Two scenarios are considered. In the homogeneous scenario, each client is considered to have identical computational capability. In the heterogeneous scenario, the computational capability varies between clients.

In the homogeneous scenario, it is assumed that the computation time of each client is τ. We also assume that channel conditions and power allocations have been omitted for simplicity. The model transmission time in the uplink is assumed to be τ^u. Model transmission in the downlink is assumed to be τ^d. The total completion time for one round in SFL is $\tau_{ho}^{syn} = \tau^d + \tau + M \cdot \tau^u$ when using TDMA. The global model updates every τ_{ho}^{syn} time. To give a fair comparison, the client can be scheduled to upload the local model again only when other clients are scheduled in AFL. Therefore, it takes $\tau_{ho}^{asyn} = M \cdot \tau^u + M \cdot \tau^d + \tau$ time for AFL to finish the same operations. AFL takes $(M-1) \cdot \tau^d$ more time than SFL due to the global model download. However, the global model is updated every $\tau^u + \tau^d$ time in AFL rather than waiting $\tau^d + \tau + M \cdot \tau^u$ time as in SFL.

In the heterogeneous scenario, the computation time for the fastest client is assumed to be τ, while the $a \cdot \tau$ time for the slowest client with $a \gg 1$. Typically, computation is faster than communication, which makes communication a bottleneck for the system. However, when slow clients also execute other tasks, $a \cdot \tau$ can be larger than $M \cdot \tau^u$. In SFL, the global model waits $\tau_{he}^{syn} = \tau^d + a \cdot \tau + M \cdot \tau^u$ time to update. The slowest client dominates the computation time. The faster clients remain idle in the waiting time. In AFL, it takes τ_{he}^{asyn} time to complete the same operations, with $M \cdot \tau^d + \tau + M \cdot \tau^u \le \tau_{he}^{asyn} \le M \cdot \tau^d + a \cdot \tau + M \cdot \tau^u$. However, the global model is updated every $\tau^u + \tau^d$ time. As shown in Fig. 7.2, the vertical dashed lines represent the global model update time. The global model gets updated more frequently in AFL than in SFL.

The time comparison is summarized in Table 7.1.

From the above analysis, the AFL does not guarantee better learning performance than the SFL. The global model is updated faster, so the learning pace is faster. However,

Table 7.1: Time Comparison

	Homogeneous Scenario	Heterogeneous Scenario
SFL	$\tau_{ho}^{syn} = \tau^d + \tau + M \cdot \tau^u$	$\tau_{he}^{syn} = \tau^d + a \cdot \tau + M \cdot \tau^u$
AFL	$\tau_{ho}^{asyn} = M \cdot \tau^u + M \cdot \tau^d + \tau$	$M \cdot \tau^d + \tau + M \cdot \tau^u \leq \tau_{he}^{asyn} \leq M \cdot \tau^d + a \cdot \tau + M \cdot \tau^u$

the local model learned from a very old starting point may have a destructive rather than constructive effect on the global model. Model staleness could hinder the global model's convergence. It is critical to have the aggregation coefficient β to control the contribution of the local model to the global model. Client scheduling also plays a key role in promoting learning convergence and providing fairness of client access.

7.3 Proposed Algorithm

In this section, the global aggregation of AFL is analyzed. First, the SFL aggregation coefficient is applied to the AFL to understand its behavior. Then, an AFL aggregation algorithm is designed to attain a learning performance comparable to that of SFL. Finally, a client scheduling and model aggregation framework in AFL (CSMAAFL) is proposed.

7.3.1 SFL Aggregation Coefficient in AFL

In SFL, the global aggregation coefficient α is predetermined and fixed in each round. It is determined by considering each client's relative number of data samples. Given a specific client scheduling $\phi(1), \phi(2), \ldots, \phi(M)$, where $\phi(i)$ is the client scheduled in global iteration i, Equation (7.3) when using SFL aggregation coefficient, can be written as:

$$\begin{aligned} w_{j+1} &= (1-\alpha_{\phi(j)})w_j + \alpha_{\phi(j)} w_i^{\phi(j)} \\ &= (1-\alpha_{\phi(j)})((1-\alpha_{\phi(j-1)})w_{j-1} + \alpha_{\phi(j-1)} w_k^{\phi(j-1)}) \\ &\quad + \alpha_{\phi(j)} w_i^{\phi(j)}. \end{aligned} \tag{7.5}$$

Here, $\alpha_{\phi(j)} = 1 - \beta_j$, $\phi(j-1)$ is the client scheduled in iteration $j-1$, and k is the last iteration when client $\phi(j-1)$ was scheduled. Similarly, the aggregation coefficient for client $\phi(j-1)$ is $\alpha_{\phi(j-1)}(1-\alpha_{\phi(j)})$. And for the first client scheduling $\phi(1)$, the aggregation coefficient is $\alpha_{\phi(1)}(1-\alpha_{\phi(2)})(1-\alpha_{\phi(3)})\cdots(1-\alpha_{\phi(j)})$. Since $\alpha \in (0,1)$, the aggregation coefficient in AFL increases over time, which violates the intuition that the contribution of the individual client decreases over time.

7.3.2 Baseline

AFL can achieve learning performance comparable to that of SFL with adequate aggregation coefficients. This requires AFL to apply the same client scheduling and aggregation coefficients. Clients can be scheduled again only when other clients are scheduled in AFL. The global iteration coefficient β also varies in each iteration. The relationship between the global iteration coefficient in SFL and AFL can be formulated as follows:

$$\sum_{m=1}^{M} \alpha_m w^m = w_{M+1} = \beta_M w_M + (1-\beta_M) w^{\phi(M)}. \tag{7.6}$$

The left-hand side (LHS) in equation (7.6) is the aggregation of the global model in SFL, while the right-hand side (RHS) corresponds to the global model after iterating all clients once in AFL. Also, the global aggregation of AFL can be further written as

$$\begin{aligned} w_M &= \beta_{M-1} w_{M-1} + (1-\beta_{M-1}) w^{\phi(M-1)} \\ &= \beta_{M-1}(\beta_{M-2} w_{M-2} + (1-\beta_{M-2}) w^{\phi(M-2)}) \\ &\quad + (1-\beta_{M-1}) w^{\phi(M-1)} \\ &= \beta_1(\dots) + (1-\beta_1) w^{\phi(1)}. \end{aligned} \tag{7.7}$$

By jointly considering Equation (7.6) and (7.7), an euqation is formulated as:

$$\alpha_{\phi(M)} = 1 - \beta_M. \tag{7.8}$$

When client scheduling $\phi(1), \phi(2), \dots, \phi(M)$ is predetermined and the aggregation coefficients α are known, β_M can be calculated. Additionally,

$$\alpha_{\phi(M-1)} = \beta_M(1-\beta_{M-1}). \tag{7.9}$$

Since β_M is now known, β_{M-1} can also be calculated. Following this approach, β_{M-2}, β_{M-3} to β_1 can be solved.

From the above analysis, we understand the behavior of AFL aggregation and establish a baseline for AFL to achieve the same learning performance as SFL.

7.3.3 CSMAAFL: Client Scheduling and Model Aggregation in AFL

In SFL, faster clients become idle when they complete their local training and wait for other clients. To take advantage of the advantages offered by AFL, we apply a client scheduling and model aggregation scheme. It considers clients' computational capabilities and client fairness. The client can request channel access to upload its local model when local training is completed. The server assigns more local iterations to clients with more computational capabilities to control the local training time. The clients report the computational capability each time the local models are uploaded. The server uses the slowest client as a reference to learn the computational capabilities of other clients. A slotted ALOHA protocol as in [102] is used for clients to request channels. With client fairness, priority is given to the client with the older model when two clients request the channel simultaneously. If client m and n apply for channel access at iteration k simultaneously, client m will grant the channel if $(k-m') > (k-n')$ where m' and n' represent the iteration client m and n access the channel the last time.

When some clients are extremely fast or slow, extra consideration is required to ensure that all clients have a fair opportunity to contribute to the global model. We employ a policy similar to [99], where faster clients perform more iterations and spend more time on local training. Now, the contributions of different clients can be balanced regardless of the computational speed.

The client scheduling problem is addressed above by considering computational capability and client fairness. The model aggregation needs to balance the current global model with the local model received in each iteration. The contribution from individual client models should diminish over time. Also, the model staleness should be considered. We use $j-i$ to represent the difference between the current iteration and the iteration when the client m was scheduled the last time. When $j-i$ is small, the local model of client m is learned from a "fresh" global model. The moving average μ_{ji} is introduced to capture the average value of $j-i$ over time. Let

$$(1-\beta_j)w_i^m = \min(1, \frac{\mu_{ji}}{\gamma j(j-i)})w_i^m, \tag{7.10}$$

for equation (7.3), where $\gamma \in (0,1)$ serves as a hyperparameter to capture the time and other factors. $\frac{1}{j}$ denotes the gradual decrease in the contribution of the individual client over time. $\frac{\mu_{ji}}{j-i}$ represents the staleness effects. When the learning starting point i of client m is recent (i.e., $j-i$ is small), the value of $\frac{\mu_{ji}}{j-i}$ will be large. The fresh model will contribute more than the stale model. The user scheduling algorithm ensures that all clients have a similar opportunity to contribute to the global model. This makes the value of $\frac{\mu_{ji}}{j-i}$ always close to 1. The stability of the system can be achieved. The complete algorithm is summarized in Algorithm 6.

Algorithm 6 Asynchronous Federated Learning with Client Scheduling and Model Aggregation

```
Initialization: Server initializes $w_0$ and broadcasts to all Clients.
while not converge do
    Client:
        Receives the most recent aggregated global model.
        Performs local computation as Eq. (7.4).
        Applies for uploading time slot.
        Upload the calculated local model and estimated local computational capacity when
    the request is approved.
    Server:
        Approves the first client $m$ requested the time slot.
        Receives the local model and computational capability from the client $m$.
        Performs aggregation by Eq. (7.6) and Eq. (7.10).
        Sends the aggregated global model and the number of local computation iterations
    to the client $m$.
end while
```

7.4 Simulation

7.4.1 Simulation Setting

This section executes the simulation of the MNIST and Fashion-MNIST datasets in both IID and non-IID data settings. We also discussed the impact of the hyperparameter λ here.

There are 100 clients connected to the server in this FL system. In SFL, all clients participate in each round's learning. For AFL, a client is scheduled again only when other clients are scheduled. To simulate the heterogeneity of the computational capabilities of clients, the selection of clients is randomized in each iteration. The convolutional neural network (CNN) with two convolutional layers, two max-pooling layers, and two fully connected layers is employed here. The learning rate η is 0.01, and the value of γ is set as 0.1, 0.2, 0.4, and 0.6, respectively. Four simulation scenarios are considered, incorporating two datasets, MNIST and Fashion-MNIST, and two data distributions, IID and non-IID.

7.4.2 Simulation Results

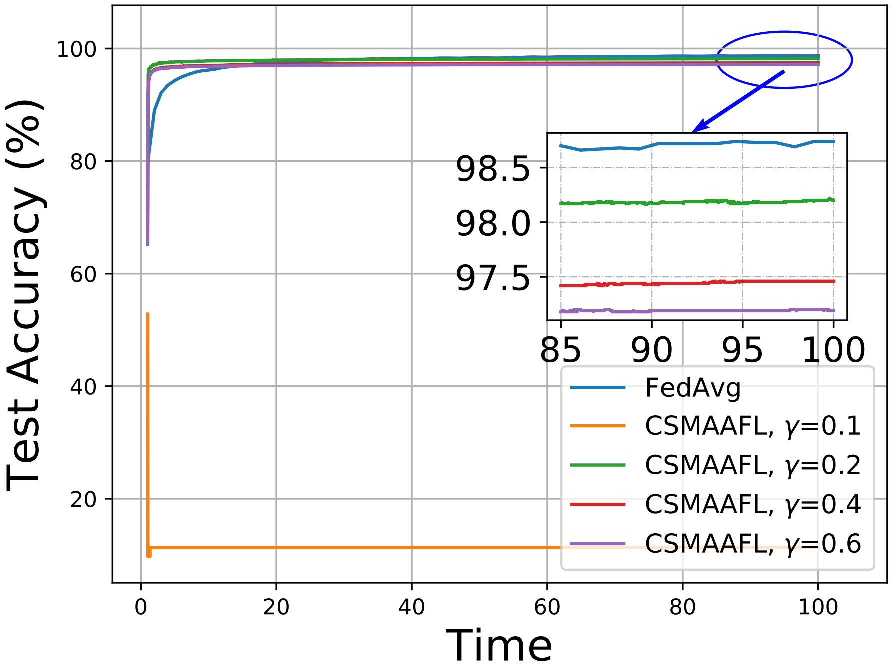


Fig. 7.3: Scenario 1: MNIST IID

In Fig. 7.5, scenario 1, MNIST with IID is considered. All schemes, except CSMAAFL with $\gamma = 0.1$, achieve similar learning performance. This indicates that the proposed CSMAAFL scheme converges and gets similar test accuracy as FedAvg when γ is appropriately tuned.

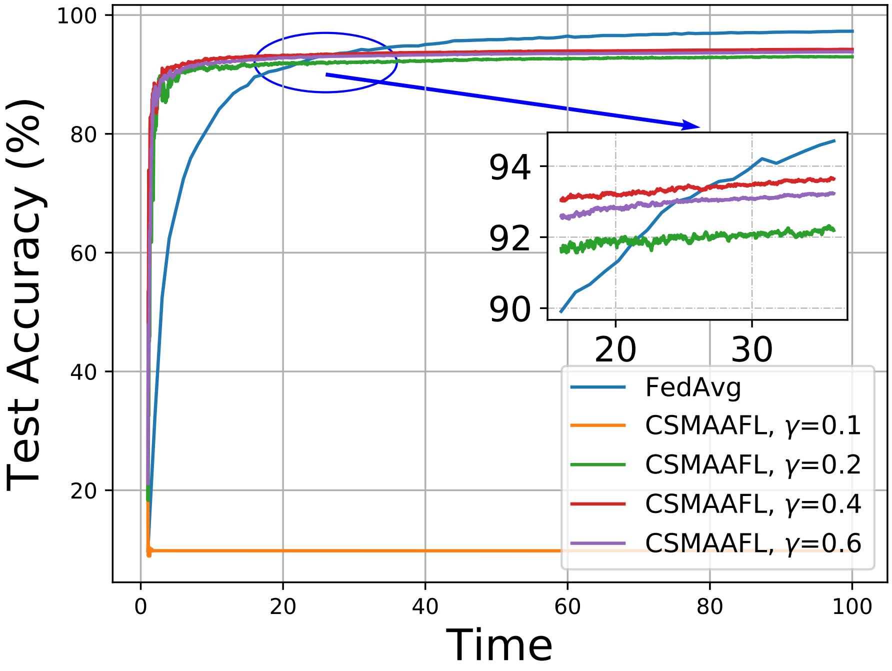


Fig. 7.4: Scenario 2: MNIST non-IID

In Fig. 7.4, MNIST with non-IID is considered. CSMAAFL learns faster than FedAvg, only after around 25 relative time slots, FedAvg approach, and beyond the CSMAAFL. This demonstrates the faster learning speed of CSMAAFL.

In Fig. 7.5, CSMAAFL with $\gamma = 0.2$ gets the closest results as FedAvg. While in Fig. 7.6, CSMAAFL with $\gamma = 0.4$ achieves the best test accuracy. It takes around 55 relative time slots for FedAvg to reach the same performance as CSMAAFL. All results demonstrate that CSMAAFL converges faster while maintaining overall learning performance.

The effect of the value of γ varies in different scenarios. In scenarios 1, 2, and 4, $\gamma = 0.1$ results in random guessing. This is because the contribution of the individual local model is over-emphasized. $\gamma = 0.2$ achieves the best test accuracy in scenarios 1 and 3 when the

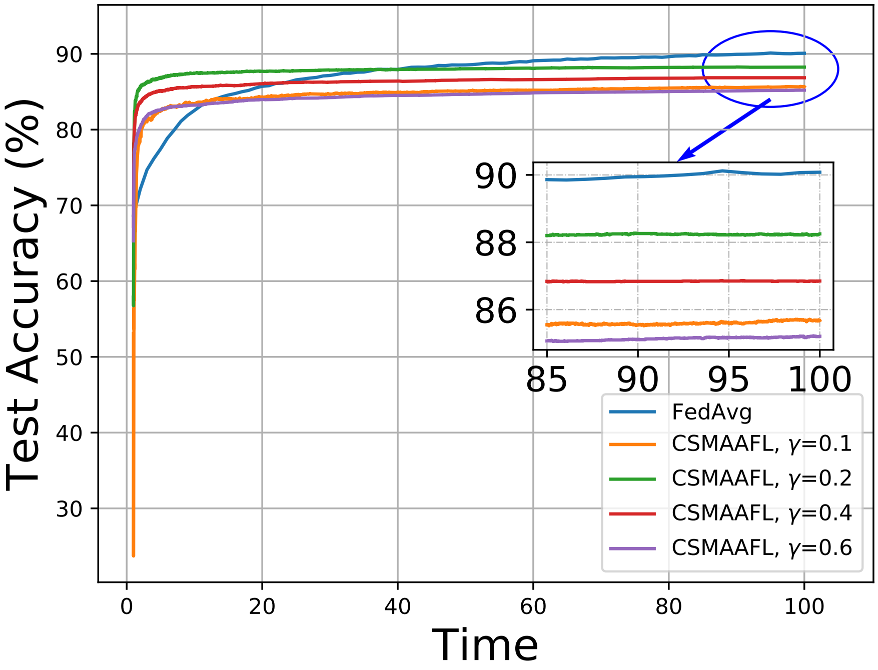


Fig. 7.5: Scenario 3: Fashion-MNIST IID

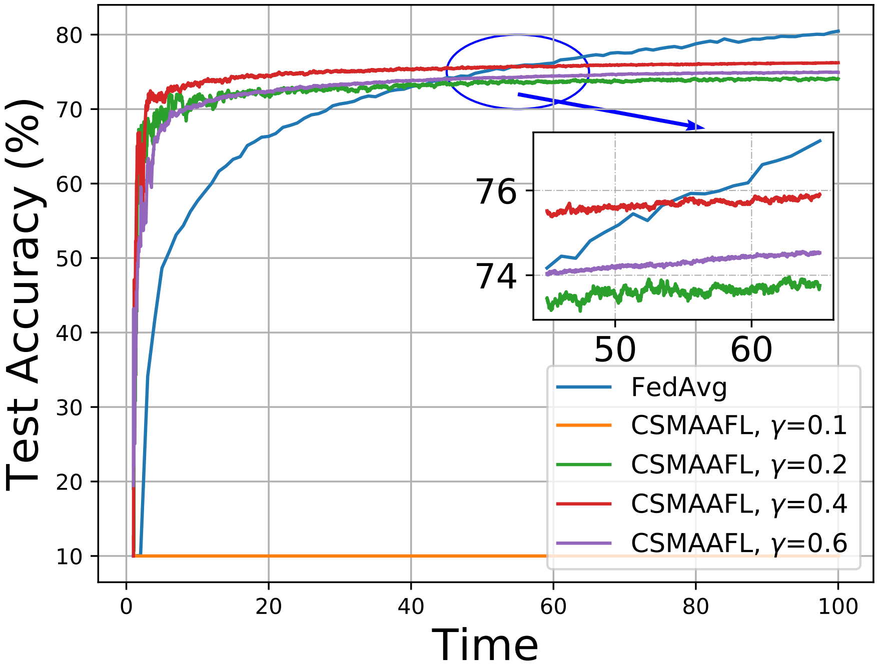


Fig. 7.6: Scenario 4: Fashion-MNIST IID

data are IID. When data is non-IID, as in Scenarios 2 and 4, $\gamma = 0.4$ achieves the best results.

7.5 Chapter Conclusion

This chapter studies system heterogeneity problems in FL. They can be addressed with asynchronous communication. In AFL, a client scheduling and model aggregation scheme is introduced to utilize computational capabilities, maintain client fairness, and solve model staleness. The simulation results demonstrate that the proposed scheme can speed up convergence while achieving a comparable final learning performance.

CHAPTER 8
Conclusions

8.1 Summary

This dissertation analyzed the motivation to apply federated learning in modern distributed learning architecture. Existing problems, including communication costs, system heterogeneity, statistical heterogeneity, security, and privacy issues, still prevent the application of FL in more fields.

Due to the application of FL on mobile devices and other wireless-connected devices, the communication costs of FL on wireless networks are first analyzed. Wireless channels are limited and expensive. It is significant for reducing communication time and improving spectral efficiency. First, we applied NOMA to the FL system to allow multiple users to simultaneously transmit their model parameters on the same channel. Gradient compression is also used to satisfy channel capacity constrained by channel conditions. Then, we employ the user scheduling and power allocation in the NOMA FL system. Joint user scheduling and power allocation help improve learning performance. We further investigated over-the-air computation in FL model transmission. It performs computations in the air, thus further reducing the computation time. Approximate communication is applied to transmit the FL model parameters approximately without forward error correction and packet retransmission. It still achieves comparable learning performance with transmission errors while the transmission time is significantly reduced.

Security and privacy issues are solved with model update-based aggregation and individual client model initiation schemes. The model update distribution is compared with the model distribution. Model update-based FL is less susceptible to Byzantine attacks. Individual client model initialization and model update can hide the local model, preventing membership interference attacks.

The system heterogeneity is addressed with a novel federated asynchronous client scheduling and model aggregation learning architecture. In each round, a single client participates in the learning process. Client scheduling considers both computational capability and user fairness. The model aggregation balances the contribution of the single-client local model and the previous global model. Asynchronous FL allows the training to run faster.

The statistical heterogeneity is considered as all of the above. The non-IID data distribution assigns the data to clients varying in sample label and number. The proposed methods also achieve good performance on non-IID data.

8.2 Future Work

The works mentioned above are utilized on clean public data sets. It might be challenging to train a converged model on practical datasets. So, practical data can be collected and trained using our proposed methods. In addition, simulations were performed to verify the effectiveness of the proposed methods. A real deployment on hardware with distributed devices can further help to demonstrate our proposed methods.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” *Artificial intelligence and statistics*, pp. 1273–1282, 2017.

[2] Ericsson, “Number of smartphone mobile network subscriptions worldwide from 2016 to 2022, with forecasts from 2023 to 2028,” Jun. 2023, [Online]. Available:https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[5] P. Voigt and A. Von dem Bussche, “The eu general data protection regulation (gdpr),” *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.

[6] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, “A review of applications in federated learning,” *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[7] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, “Federated learning of out-of-vocabulary words,” *arXiv preprint arXiv:1903.10635*, 2019.

[8] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li, “Pmf: A privacy-preserving human mobility prediction framework via federated learning,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–21, 2020.

[9] B. Hu, Y. Gao, L. Liu, and H. Ma, “Federated region-learning: An edge computing based framework for urban environment sensing,” in *2018 ieee global communications conference (globecom)*. IEEE, 2018, pp. 1–7.

[10] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, “Federated learning-based cognitive detection of jamming attack in flying ad-hoc network,” *IEEE Access*, vol. 8, pp. 4338–4350, 2019.

[11] G. Szegedi, P. Kiss, and T. Horváth, “Evolutionary federated learning on eeg-data.” in *ITAT*, 2019, pp. 71–78.

[12] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu, “Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records,” *Journal of biomedical informatics*, vol. 99, p. 103291, 2019.

[13] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[14] L. Liu, J. Zhang, S. Song, and K. B. Letaief, “Client-edge-cloud hierarchical federated learning,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[15] H. Xing, O. Simeone, and S. Bi, “Decentralized federated learning via sgd over wireless d2d networks,” in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.

[16] A. F. Aji and K. Heafield, “Sparse communication for distributed gradient descent,” *arXiv preprint arXiv:1704.05021*, 2017.

[17] X. Fan, Y. Wang, Y. Huo, and Z. Tian, “Communication-efficient federated learning through 1-bit compressive sensing and analog aggregation,” in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2021, pp. 1–6.

[18] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.

[19] H. Sun, X. Ma, and R. Q. Hu, “Adaptive federated learning with gradient compression in uplink noma,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 325–16 329, 2020.

[20] K. Yang, T. Jiang, Y. Shi, and Z. Ding, “Federated learning via over-the-air computation,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[21] H. H. Yang, Z. Liu, T. Q. Quek, and H. V. Poor, “Scheduling policies for federated learning in wireless networks,” *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, 2019.

[22] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, “Joint device scheduling and resource allocation for latency constrained wireless federated learning,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2020.

[23] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani, “Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity,” *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 2, pp. 197–205, 2022.

[24] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. V. Poor, “Convergence of update aware device scheduling for federated learning at the wireless edge,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3643–3658, 2021.

[25] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, “Asynchronous online federated learning for edge devices with non-iid data,” in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 15–24.

[26] T. Chen, X. Jin, Y. Sun, and W. Yin, “Vafl: a method of vertical asynchronous federated learning,” *arXiv preprint arXiv:2007.06081*, 2020.

[27] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” *arXiv preprint arXiv:1806.00582*, 2018.

[28] Y. Deng, M. M. Kamani, and M. Mahdavi, “Adaptive personalized federated learning,” *arXiv preprint arXiv:2003.13461*, 2020.

[29] J. So, B. Güler, and A. S. Avestimehr, “Byzantine-resilient secure federated learning,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[30] C. Wu, X. Yang, S. Zhu, and P. Mitra, “Mitigating backdoor attacks in federated learning,” *arXiv preprint arXiv:2011.01767*, 2020.

[31] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, “Differential privacy-enabled federated learning for sensitive health data,” *arXiv preprint arXiv:1910.02578*, 2019.

[32] M. Seif, R. Tandon, and M. Li, “Wireless federated learning with local differential privacy,” in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 2604–2609.

[33] S. C. Ergen and P. Varaiya, “Tdma scheduling algorithms for wireless sensor networks,” *Wireless networks*, vol. 16, pp. 985–997, 2010.

[34] H. G. Myung, J. Lim, and D. J. Goodman, "Single carrier fdma for uplink wireless transmission," *IEEE vehicular technology magazine*, vol. 1, no. 3, pp. 30–38, 2006.

[35] Y. Saito, Y. Kishiyama, A. Benjebbour, T. Nakamura, A. Li, and K. Higuchi, "Non-orthogonal multiple access (noma) for cellular future radio access," in *2013 IEEE 77th vehicular technology conference (VTC Spring)*. IEEE, 2013, pp. 1–5.

[36] Z. Zhang, H. Sun, and R. Q. Hu, "Downlink and uplink non-orthogonal multiple access in a dense wireless network," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 12, pp. 2771–2784, 2017.

[37] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[38] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[39] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020.

[40] N. Zhang, J. Wang, G. Kang, and Y. Liu, "Uplink nonorthogonal multiple access in 5g systems," *IEEE Communications Letters*, vol. 20, no. 3, pp. 458–461, 2016.

[41] M. M. Al-Wani, A. Sali, B. M. Ali, A. A. Salah, K. Navaie, C. Y. Leow, N. K. Noordin, and S. J. Hashim, "On short term fairness and throughput of user clustering for downlink non-orthogonal multiple access system," in *2019 IEEE 89th vehicular technology conference (VTC2019-Spring)*. IEEE, 2019, pp. 1–6.

[42] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[43] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” *arXiv preprint arXiv:1606.06160*, 2016.

[44] A. Chandra and K. Chakrabarty, “System-on-a-chip test-data compression and decompression architectures based on golomb codes,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 3, pp. 355–368, 2001.

[45] Y. LeCun, C. Cortes, and C. Burges, “The mnist database of handwritten digits,” 1998, [Online]: http://yann.lecun.com/exdb/mnist/.

[46] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, “Leaf: A benchmark for federated settings,” *arXiv preprint arXiv:1812.01097*, 2018. [Online]. Available: https://arxiv.org/abs/1812.01097

[47] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[48] A. Douik, H. Dahrouj, T. Y. Al-Naffouri, and M.-S. Alouini, “Coordinated scheduling and power control in cloud-radio access networks,” *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2523–2536, 2015.

[49] L. P. Qian, Y. J. Zhang, and J. Huang, “Mapel: Achieving global optimality for a non-convex wireless power control problem,” *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1553–1563, 2009.

[50] D. Zhai and J. Du, “Spectrum efficient resource management for multi-carrier-based noma networks: A graph-based method,” *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 388–391, 2017.

[51] Wikipedia, “Independent set (graph theory),” Feb. 2024, [Online]. Available:https://en.wikipedia.org/wiki/Independent_set_(graph_theory).

[52] M. R. Garey and D. S. Johnson, "'strong' np-completeness results: Motivation, examples, and implications," *Journal of the ACM (JACM)*, vol. 25, no. 3, pp. 499–508, 1978.

[53] A. Alexiou, C. Bouras, and A. Papazois, "Adopting forward error correction for multicasting over cellular networks," in *2010 European Wireless Conference (EW)*. IEEE, 2010, pp. 361–368.

[54] M.-F. Tsai, N. Chilamkurti, C.-K. Shieh, and A. Vinel, "Mac-level forward error correction mechanism for minimum error recovery overhead and retransmission," *Mathematical and computer modelling*, vol. 53, no. 11-12, pp. 2067–2077, 2011.

[55] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53 841–53 849, 2020.

[56] A. Nafaa, T. Taleb, and L. Murphy, "Forward error correction strategies for media streaming over wireless networks," *IEEE Communications Magazine*, vol. 46, no. 1, pp. 72–79, 2008.

[57] M. Shirvanimoghaddam, A. Salari, Y. Gao, and A. Guha, "Federated learning with erroneous communication links," *IEEE communications letters*, vol. 26, no. 6, pp. 1293–1297, 2022.

[58] X. Su, Y. Zhou, L. Cui, and J. Liu, "On model transmission strategies in federated learning with lossy communications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 4, pp. 1173–1185, 2023.

[59] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17 322–17 341, 2017.

[60] Z. Du, K. Palem, A. Lingamneni, O. Temam, Y. Chen, and C. Wu, "Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators," in *2014 19th Asia and South Pacific design automation conference (ASP-DAC).* IEEE, 2014, pp. 201–206.

[61] F. Betzel, K. Khatamifard, H. Suresh, D. J. Lilja, J. Sartori, and U. Karpuzcu, "Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–32, 2018.

[62] A. M Abdelmoniem, A. Elzanaty, M.-S. Alouini, and M. Canini, "An efficient statistical-based gradient compression technique for distributed training systems," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 297–322, 2021.

[63] B. Guo, Y. Liu, and C. Zhang, "A partition based gradient compression algorithm for distributed training in aiot," *Sensors*, vol. 21, no. 6, p. 1943, 2021.

[64] F. P. Sunny, A. Mirza, I. Thakkar, M. Nikdast, and S. Pasricha, "Arxon: A framework for approximate communication over photonic networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 6, pp. 1206–1219, 2021.

[65] Y. Chen and A. Louri, "An approximate communication framework for network-on-chips," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1434–1446, 2020.

[66] M. F. Reza and P. Ampadu, "Approximate communication strategies for energy-efficient and high performance noc: Opportunities and challenges," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, 2019, pp. 399–404.

[67] S. Sen, S. Gilani, S. Srinath, S. Schmitt, and S. Banerjee, "Design and implementation of an" approximate" communication system for wireless media applications," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 15–26.

[68] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *International conference on machine learning.* PMLR, 2014, pp. 1000–1008.

[69] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational vision science & technology*, vol. 9, no. 2, pp. 14–14, 2020.

[70] M. A. Nielsen, *Neural networks and deep learning.* Determination press San Francisco, CA, USA, 2015, vol. 25.

[71] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[72] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[73] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning.* PMLR, 2013, pp. 1139–1147.

[74] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[75] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[76] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[77] P. Murugan, "Feed forward and backward run in deep convolution neural network," *arXiv preprint arXiv:1711.03278*, 2017.

[78] G. Philipp, D. Song, and J. G. Carbonell, "The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions," *arXiv preprint arXiv:1712.05577*, 2017.

[79] H. H. Tan and K. H. Lim, "Vanishing gradient mitigation with deep learning neural network optimization," in *2019 7th international conference on smart computing & communications (ICSCC)*. IEEE, 2019, pp. 1–4.

[80] Y. Chen and A. Louri, "Learning-based quality management for approximate communication in network-on-chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3724–3735, 2020.

[81] M. Duarte, A. Sabharwal, V. Aggarwal, R. Jana, K. K. Ramakrishnan, C. W. Rice, and N. Shankaranarayanan, "Design and characterization of a full-duplex multiantenna system for wifi networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 3, pp. 1160–1177, 2013.

[82] B. K. Butler, "Minimum distances of the qc-ldpc codes in ieee 802 communication standards," *arXiv preprint arXiv:1602.02831*, 2016.

[83] S. Varma, "Chapter 4 - congestion control in broadband wireless networks," *Internet Congestion Control*, pp. 103–134. [Online]. Available: https://doi.org/10.1016/B978-0-12-803583-2.00004-9

[84] X. Cao, G. Zhu, J. Xu, and K. Huang, "Optimized power control for over-the-air computation in fading channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7498–7513, 2020.

[85] X. Zang, W. Liu, Y. Li, and B. Vucetic, "Over-the-air computation systems: Optimal design with sum-power constraint," *IEEE Wireless Communications Letters*, vol. 9, no. 9, pp. 1524–1528, 2020.

[86] C. Xu, S. Liu, Z. Yang, Y. Huang, and K.-K. Wong, "Learning rate optimization for federated learning exploiting over-the-air computation," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3742–3756, 2021.

[87] L. Chen, X. Qin, and G. Wei, "A uniform-forcing transceiver design for over-the-air function computation," *IEEE Wireless Communications Letters*, vol. 7, no. 6, pp. 942–945, 2018.

[88] A. Konar and N. D. Sidiropoulos, "Fast approximation algorithms for a class of non-convex qcqp problems using first-order methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3494–3509, 2017.

[89] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[90] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[91] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[92] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[93] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[94] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[95] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data. arxiv 2016," *arXiv preprint arXiv:1611.04482*.

[96] A. Elgabli, J. Park, C. B. Issaid, and M. Bennis, "Harnessing wireless channels for scalable and privacy-preserving federated learning," *IEEE Transactions on Communications*, vol. 69, no. 8, pp. 5194–5208, 2021.

[97] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[98] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE, 2019, pp. 1–7.

[99] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[100] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 21–28.

[101] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[102] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. Quek, "Asynchronous federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6961–6978, 2022.

[103] Q. Wang, Q. Yang, S. He, Z. Shi, and J. Chen, "Asyncfeded: Asynchronous federated learning with euclidean distance based adaptive weight aggregation," *arXiv preprint arXiv:2205.13797*, 2022.

CURRICULUM VITAE

Xiang Ma

Xiang Ma received a Bachelor's degree from the Southern University of Science and Technology, Shenzhen, China, in 2017 under the supervision of Prof. Jin Zhang. After that, he worked as a software engineer at Sangfor Technologies. He is currently pursuing a Ph.D. degree under the supervision of Prof. Rose Qingyang Hu with the Department of Electrical and Computer Engineering at Utah State University, Logan, UT.

Teaching Assistant Experience

- ECE 5600: Introduction to Computer Networks, Fall 2020, Fall 2021, Fall 2022, Fall 2023.
- ECE 6600: Wireless and Mobile Networking, Spring 2021, Spring 2022, Spring 2023.

Journal Articles

- Approximate Wireless Communication for Lossy Gradient Updates in Federated Learning, **Xiang Ma**, Haijian Sun, Rose Qingyang Hu, and Yi Qian, Submitted to *IEEE Internet of Things Journal* in April, 2024.
- Exploring Communication Technologies, Standards, and Challenges in Electrified Vehicle Charging, **Xiang Ma**, Yuan Zhou, Hanwen Zhang, Qun Wang, Haijian Sun, Hongjie Wang, and Rose Qingyang Hu, Accepted by *IET Communications Journal* in May 2024.
- Precise Coil Alignment for Dynamic Wireless Charging of Electric Vehicles with RFID Sensing, Haijian Sun, **Xiang Ma**, Rose Qingyang Hu, and Randy Christensen, Submitted to *IEEE Wireless Communication Magazine* in Decemeber, 2023.

- Adaptive federated learning with gradient compression in uplink NOMA, Haijian Sun, **Xiang Ma**, and Rose Qingyang Hu, *IEEE Transactions on Vehicular Technology*, vol. 69, no.12, pp. 16325-16329, 2020.

Conference Papers

- CSMAAFL: Client Scheduling and Model Aggregation in Asynchronous Federated Learning, **Xiang Ma**, Haijian Sun, Rose Qingyang Hu, and Yi Qian, in *IEEE International Conference on Communications (ICC)*, 2024.

- Energy-Efficient Secure Offloading for NOMA-Enabled Machine-type Mobile-Edge Computing, Yuan Zhou, Haijian Sun, **Xiang Ma**, and Rose Qingyang Hu, *2023 IEEE International Conference on Industrial Technology (ICIT)*, 2023.

- Approximate Wireless Communication for Federated Learning, **Xiang Ma**, Haijian Sun, Rose Qingyang Hu, and Yi Qian, in *ACM Workshop on Wireless Security and Machine Learning (WiseML 2023)*, 2023.

- A New Implementation of Federated Learning for Privacy and Security Enhancement, **Xiang Ma**, Haijian Sun, Rose Qingyang Hu, and Yi Qian, *IEEE Global Communications Conference (GLOBECOM)*, 2022.

- User Scheduling for Federated Learning Through Over-the-Air Computation, **Xiang Ma**, Haijian Sun, Qun Wang, and Rose Qingyang Hu, in *IEEE IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 2021.

- Scheduling policy and power allocation for federated learning in NOMA based MEC, **Xiang Ma**, Haijian Sun, and Rose Qingyang Hu, in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

- Towards green mobile edge computing offloading systems with security enhancement, Haijian Sun, Qun Wang, **Xiang Ma**, Yongjun Xu, and Rose Qingyang Hu, in *2020 Intermountain Engineering, Technology and Computing (IETC)*, 2020.

Honors and Awards

- IEEE ICC NSF travel award, 2022.
- Excellent New Employee in Sangfor Technologies, 2017
- Newly-Established Scholarship from SUSTech, 2013-2016

Academic Service

- Invited TPC Member:
  - 2023 International Conference on Wireless Communications and Signal Processing (WCSP)
- Invited Reviewer for Conferences:
  - IEEE International Conference on Computer Communications (INFOCOM) (2024)
  - IEEE Conference on Vehicular Technology (VTC) (2021 Fall, 2023 Fall, 2024 Spring)
  - IEEE International Conference on Communications in China (ICCC) (2021)
- Invited Reviewer for Journals:
  - IEEE Transactions on Vehicular Technology
  - IEEE Wireless Communications Magazine
  - IEEE Vehicular Technology Magazine
  - IEEE Transactions on Network Science and Engineering
  - IEEE Transactions on Emerging Telecommunications Technologies