

**COMMUNICATION-EFFICIENT
FEDERATED LEARNING FOR PERSONAL
ASSISTANT APPLICATION ON
SMARTPHONES**

BY

ABDULMUMIN SA'AD

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
COMPUTER ENGINEERING

MAY 2024

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ABDULMUMIN SA'AD** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER ENGINEERING**.

Thesis Committee

Dr. Mohaminad Abdurazzak Felemban
(Adviser)

Prof. Tarek Rahil Sheltami (Member)

Dr. Abdulaziz Salah Tabbakh (Member)

Prof. Aiman El-Maleh
Department Chairman

Prof. Soliman S. Al-Homidan

Dean of Graduate Studies

02/07/2024

Date



©Abdulmumin Sa'ad
2024

*I dedicate this thesis to my family, especially my newborn princess
{Rahma}.*

ACKNOWLEDGMENTS

I thank Almighty Allah for His infinite mercy and blessings. Alhamdulillah! I reached this milestone only by the will of Allah. I am deeply grateful to my advisor Dr. Muhamad Felemban. You were more than an advisor and your patience was priceless. You kept your door always opened for me for both physical and remote interaction, this really made it easy for me to achieve this milestone. Also, I would like to use this opportunity to express my gratitude to Dr. Abdulaziz Salah Tabbakh and Prof. Tarek Rahil Sheltami of my thesis committee for their unwavering support despite their busy schedules, I appreciate their efforts. My gratitude goes to King Fahd University of Petroleum and Minerals and the Kingdom of Saudi Arabia for providing me the opportunity to pursue master's degree under a fully-funded scholarship. In addition, I want to express my gratitude to my family for their special and unconditional support, which has been a source of strength for me throughout my academic career. In addition, I would like to thank the Nigerian Community University of Petroleum and Minerals (NCUPM) for making my stay at the university filled with so much fun. Lastly, to my newborn princess {Rahma}, your arrival made me stronger and motivated me to keep moving. Thank you all.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vii
LIST OF TABLES	x
LIST OF FIGURES	xi
ABSTRACT (ENGLISH)	xv
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement	4
1.2 Research Objectives	5
1.3 Thesis Organization	5
CHAPTER 2 BACKGROUND AND RELATED WORK	6
2.1 Personal Assistant Applications	6
2.2 Deep Learning Models for Personal Assistant Applications	8
2.3 Federated Learning	10
2.3.1 State-of-the-Art Algorithms in FL	11
2.3.2 Security and Privacy Challenges in FL	15
2.3.3 Communication-Efficient FL	17
2.4 Federated Learning for Personal Assistant Applications	20
CHAPTER 3 CEFEL FRAMEWORK	23
3.0.1 Application Development	23
3.0.2 Communication-Efficient FL via Model Parameter Freezing	25

3.1	Freezing Techniques	28
3.1.1	FedFreeze	28
3.1.2	FedFreeze+	31
3.2	CEFEL Framework	35
CHAPTER 4 PERFORMANCE EVALUATION		37
4.1	Dataset	38
4.2	Experimental Setup	40
4.2.1	Model Architecture	40
4.2.2	Hyperparameters Tuning	40
4.2.3	Ablation Study	41
4.3	Evaluation Results	42
4.3.1	Communication Cost	43
4.3.2	Computation Overhead	44
4.3.3	Model Accuracy	45
4.4	Model Deployment	46
4.5	Results Summary	48
CHAPTER 5 CONCLUSION		50
REFERENCES		52
VITAE		66

LIST OF TABLES

4.1	Experiment Hyperparameters.	41
4.2	Breakdown of Model Parameters by Layer.	41
4.3	Average Computation Cost Per Round in Seconds (s) Between CEFEL and FedAvg for non-IID Distribution and $E = 5$	42

LIST OF FIGURES

1.1	<i>Federated Learning Architecture</i>	2
2.1	<i>Keyphrase Extraction Structure [1].</i>	7
2.2	<i>Centralized Learning Architecture [2].</i>	11
2.3	<i>Federated Averaging Architecture [3].</i>	12
2.4	Communication Bottleneck in Federated Learning [4].	18
3.1	Proposed Intelligent Personal Assistant Utilizing CEFEL Framework. .	24
3.2	Speech-to-Text Application.	25
3.3	Freezing Demonstration of 8 Trainable Layers for 10 Training Rounds with (a) FedFreeze Architecture. (b) FedFreeze+ Architecture.	26
3.4	CEFEL Workflow (a) FedFreeze. (b) FedFreeze+.	28
4.1	Dataset Partitioning across 5 Clients Based on Vocabulary List for (a) IID Distribution. (b) Non-IID Distribution.	38
4.2	Experiments for selecting the best hyperparameters on non-IID Data for (a) accuracy of Best-Model Learning Rate (b) training Time of Best-Model Learning Rate (c) accuracy of Different Number of Local Epoch (d) training Time of Different Number of Local Epoch.	39
4.3	Ablation Study for (a) Convergence Curves of Model Layers (b) Ac- curacy with Frozen Layer (c) Training Time with Frozen Layer (d) Communication Cost with Frozen Layer	43

4.4	Comparison of Communication Overhead Against Number of Clients for (a) Existing Frameworks on 100 Rounds for the CIFAR-10 with non-IID using CNN [5]. (b) GUM Datasets with non-IID on CEFEL Framework for 100 Rounds using RNN.	44
4.5	(a) Training Accuracy and (b) Training Time Comparison Between Centralized and FL Setup against Local Epoch of $E = 100$ for Centralized and Communication Rounds of $E = 1$, $R = 100$ for FL.	44
4.6	Comparison Between FedAvg, FedFreeze & FedFreeze+ on (a) Test Accuracy (b) Communication Cost (c) Training Time for 10 clients with IID and non-IID Distribution.	46
4.7	Comparison Between FedAvg, FedFreeze & FedFreeze+ on (a) Test Accuracy (b) Communication Cost (c) Training Time for 30 clients with IID and non-IID Distribution.	47
4.8	<i>Keyphrase Extraction Demonstration on a Personal Assistant Application.</i>	48
4.9	(a) Model Prediction and Context Handling. (b) Model Prediction and Wrong Entry Handling.	48

THESIS ABSTRACT

NAME: Abdulmumin Sa'ad

TITLE OF STUDY: Communication-Efficient Federated Learning for Personal Assistant Application on Smartphones

MAJOR FIELD: Computer Engineering

DATE OF DEGREE: May 2024

Smartphones have rapidly become ubiquitous, with over five billion users globally. Equipped with advanced sensors such as GPS, cameras, and microphones, they generate large amounts of diverse data. Much of this data, including locations, call logs, images, and online activities, are highly private. A suitable model training architecture is required to enhance usability through intelligent applications while preserving data privacy and security. In this thesis, we develop a Communication Efficient Federated Learning (CEFEL) framework for smartphone personal assistant applications to protect user privacy and maintain user experience with low communication and computational overhead. Our framework identifies parameters that converge early and can be frozen or communicated intermittently, reducing communication overhead without compromising model accuracy. Extensive experiments demonstrate the robust-

ness of our approach against unbalanced and non-IID data distributions. The experiments show that our developed framework of FedFreeze and FedFreeze+ techniques outperforms state-of-the-art FL algorithms like FedAvg in reducing communication and computational costs, preserving privacy, and maintaining training efficiency with comparable accuracy.

CHAPTER 1

INTRODUCTION

Smartphones are becoming a necessary technology in our daily lives [6], [7]. In addition to their mobility, the advanced sensors on mobile phones, including GPS, cameras, and microphones, provide access to an unprecedented amount of diverse data. Much of the data collected through these sensors is considered private data, such as locations, phone call logs, images, and online activities. Machine learning (ML) has the potential to improve the usability of intelligent mobile applications through models trained using such data. Such models can be used to build various context-sensitive personalized systems [8], [9], [10], [11], [12].

However, implementing such ML models for personalized applications on smartphones is challenging for several reasons [6], [2], [13]. One significant challenge lies in training large ML models with huge amounts of data on resource-constrained devices. Due to the nature of applications, trained models need to be continuously updated to adapt to dynamic user behaviors and preferences [6]. Another challenge is to ensure data privacy and security when using sensitive user data to train ML models [14], [15]. Addressing these challenges requires combining recent advances in ML and privacy-

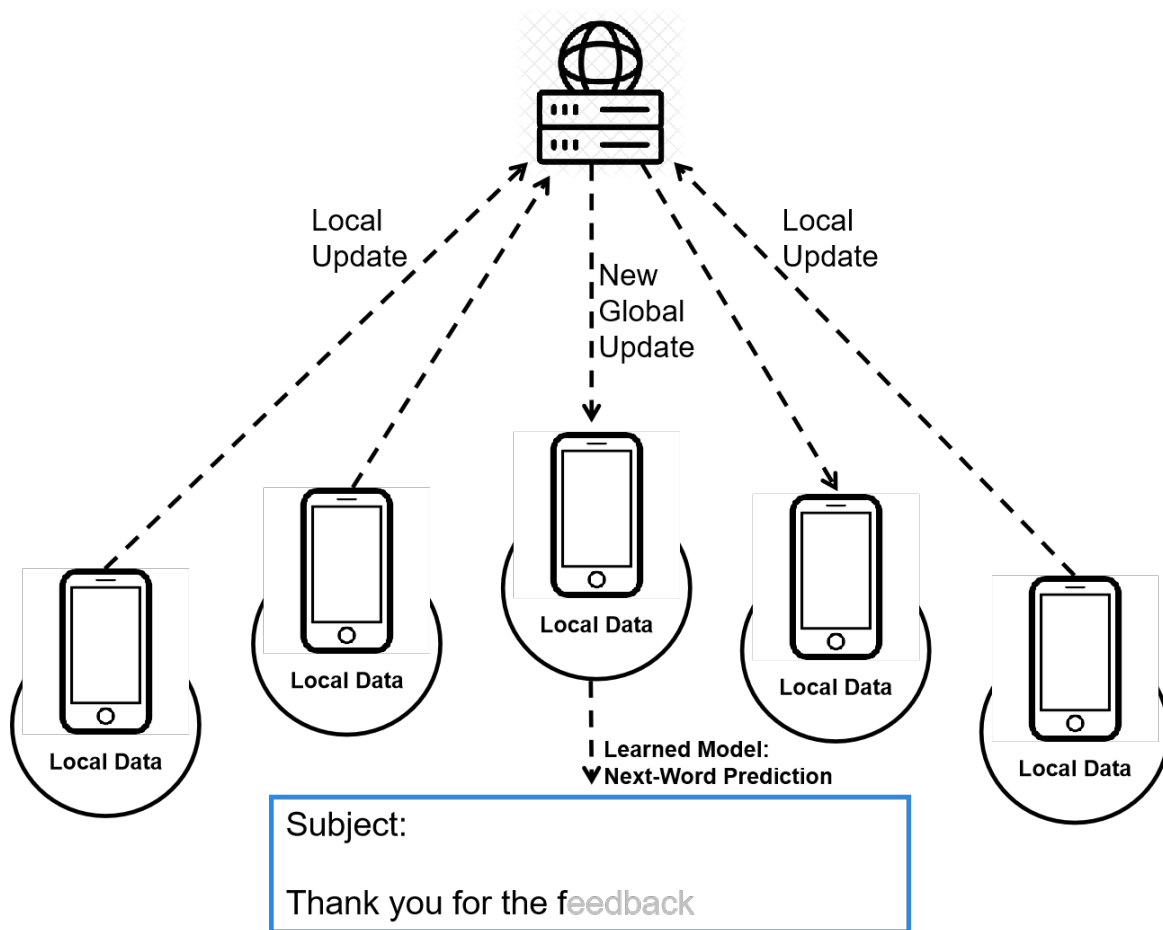


Figure 1.1: *Federated Learning Architecture*

preserving algorithms [16]. In centralized ML models, both training and inference occur on a centralized server [17], with mobile clients communicating their sensitive data with the server. This is challenging due to user privacy [18], bandwidth limitations [19], resource requirements, and efficiency [20]. The above challenges can be addressed using a decentralized learning approach such as Federated Learning (FL) [21], [22], [23].

Figure 1.1 illustrates the essence of FL, in which a dispersed collection of devices (clients) participate in the learning process by sharing their local models with a central server. The central server maintains a globally updated model. In this way, clients

do not share their confidential data during the training process. Instead, each client only needs to share a locally optimized model with the central server, which in turn aggregates the individual model by averaging [24]. The aggregated updates are then applied to improve the global model before moving on to the next training iteration, where the clients repeat the process based on the newly updated global model [21]. A prominent example of applications using FL is the GBoard text entry history [14] and activity scheduling [6]. This distributed learning framework enables personalized learning on diverse datasets without compromising user data privacy, making it particularly beneficial for data-sensitive applications such as predictive text [14], personalized recommendations [6], and healthcare care [15].

However, communication is a critical bottleneck in FL [4][21]. In particular, since the Internet connections between the clients and the server have limited capacity [19], the synchronization of the update of the model is an intrinsic bottleneck that significantly prolongs the training process. Numerous studies have been conducted to reduce the communication overhead associated with distributed model training. For example, reducing the model size by quantization into fewer bits before transmission [25].

Furthermore, several further publications [21], [23] propose to increase the amount of local computation while decreasing the total communication frequency to allow more local updates each communication round. These methods need powerful computing resources, and because local training takes precedence, there may be client-to-client differences in the global model’s performance.

To address the aforementioned challenges, Chen et al. [26] proposed a freezing

technique to reduce communication overhead by avoiding synchronizing unnecessary parameters without compromising the accuracy of the model. This involved solving two challenges: determining the converged parameters in the local model of participating clients and assessing the temporary convergence of the selected parameters. However, this work suffers from a drawback in which the performance gain would be rather little if the convergence of the parameters of the supposed layer to be frozen happened later in the training phase or never at all.

1.1 Problem Statement

In this thesis, we focus on developing a personal assistant application that analyzes phone calls and text entries to identify action keywords. For example, the proposed application would analyze the sentence 'My assistant Alice will send you our new proposal today' and create a notification/alarm reminder accordingly. Several applications exist in the literature [17], [9]. However, we emphasize on developing two features: continuous learning to enrich the model generalization to achieve optimal personalization based on each client's preferences and domain of operation while ensuring low communication and computational overheads.

The objective of this thesis is to develop and implement a Communication-Efficient Federated Learning (CEFEL) framework to support continuous model updates of the personal assistant application. The proposed model ensures the privacy of sensitive data while achieving high accuracy for the global model with less communication overhead.

1.2 Research Objectives

The objectives of this thesis are the following.

1. Design and implement CEFEL framework with the objective of minimizing the communication overhead while ensuring user data privacy.
2. Develop a personalized smartphone assistant application that uses CEFEL framework for training ML models with focus on non-iid distribution.
3. Conduct experiments to evaluate the performance of the developed framework and mobile application.

1.3 Thesis Organization

The remaining chapters of this thesis are as follows. Chapter 2 presents the background concepts and related work of this thesis. Chapter 3 presents the proposed framework and development stages of the personal assistant application and model. Chapter 4 presents the experimental setup and various performance metrics of the model. Finally, the conclusion and future work are presented in chapter 5.

CHAPTER 2

BACKGROUND AND RELATED WORK

This section highlights the background and related work on the problem. First, we present the general architecture of personal assistant applications. Then, we discuss concepts related to speech and text prediction models. Next, we highlight the state-of-the-art work related to federated learning. Finally, we present the state of the art related to the use of federated learning in personal assistant applications.

2.1 Personal Assistant Applications

Personal assistant applications are programs that can understand, respond to spoken inputs or gestures, and process the user’s request [6], [10]. In recent years, smartphones have experienced tremendous growth, going from basic calling devices to full-fledged computing devices [27]. If smartphones could recognize and adjust to their users’ behavior and carry out tasks without the user’s assistance, they would be more user-

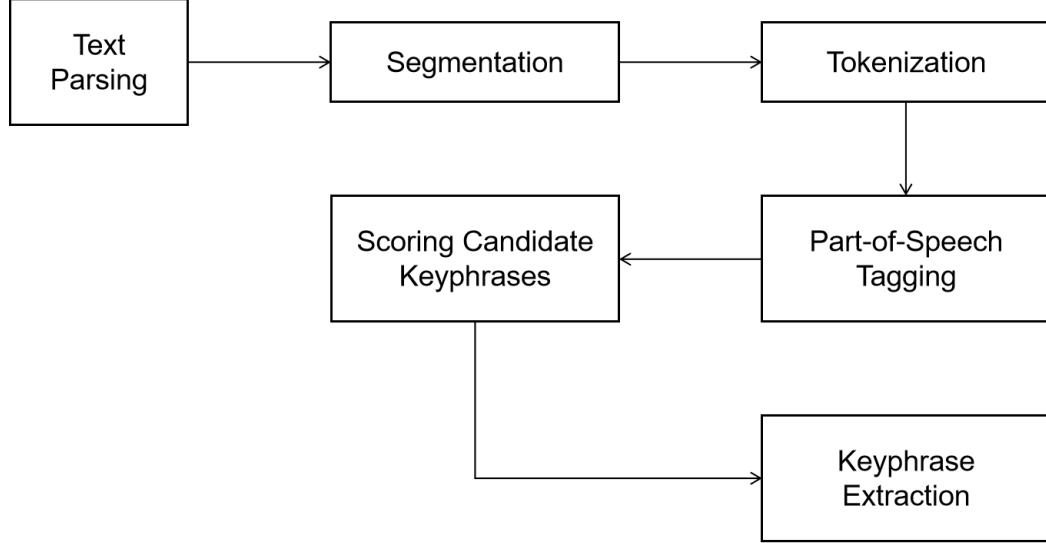


Figure 2.1: *Keyphrase Extraction Structure [1].*

friendly and enhance the user experience as in [14], [15], [11]. Figure 2.1 illustrates the various stages of the personal assistant application to obtain keyphrase extraction.

Speech-to-Text (STT) converts spoken language into written text, enabling the system to process audio inputs for keyphrase extraction. STT systems often use deep learning models, such as CNNs and RNNs, to transcribe audio data accurately [8].

Part-of-Speech (POS) Tagging is the process of labeling words in a sentence with their corresponding part-of-speech tags, such as noun, verb, adjective, etc. It helps in understanding the grammatical structure of sentences, which is crucial for accurate keyphrase extraction [8].

Keyphrase Vectorizer is a Python tool that creates a document-keyphrase matrix by extracting keyphrases with part-of-speech patterns from a set of text documents [1]. The POS patterns represent the rule-based-logic for any pattern of phrases to be extracted. A mathematical matrix that indicates the frequency of keyphrases in a set of documents is called a document-keyphrase matrix. Text documents are indicated

by the rows of the matrix, while unique keyphrases are shown by the columns. In order to calculate document-keyphrase matrices, the vectorizer extracts keyphrases from text documents using part-of-speech tags.

2.2 Deep Learning Models for Personal Assistant Applications

Machine learning encompasses the creation of algorithms and models that empower machines to learn from data. Services in personal assistant applications that are intelligent and context aware often rely on the use of ML-based prediction models that use relevant contextual user data, such as phone log data [17], [12], [28]. In this thesis, we focus on keyphrase extraction from speech and text data generated by a smartphone user using language models trained with a large annotated English corpus. However, ML models were shown to be insufficient in handling the complexity and nuances of language data without extensive manual feature engineering [29], [30], [31].

Deep learning models, on the other hand, have been shown to achieve superior performance in understanding and extracting meaningful information from text compared to traditional machine learning [6]. In particular, Recurrent Neural Network (RNN) and Long-Short-Term Memory (LSTM) have been used for automatic learning of intricate patterns and dependencies in sequential text data [31], [32]. RNN is a neural network designed for tasks that involve sequential or time series data. Unlike feedforward neural networks, RNNs have connections that form directed cycles,

allowing them to capture dependencies and patterns over time [32]. On the other hand, LSTM is a type of RNN architecture that is specifically designed to address the limitations of traditional RNNs in capturing long-range dependencies and handling vanishing gradient problems. LSTM networks have become the cornerstone in various tasks of natural language processing (NLP), including language modeling, text generation, sentiment analysis, machine translation, and keyphrase extraction, due to their ability to effectively model sequential data [32]. In particular, LSTM models capture temporal dependencies in speech, allowing the model to understand speech patterns over time [32].

RNN and LSTM have been shown to significantly improve the precision and efficiency of keyphrase extraction, as demonstrated in various and recent literature [1], [33], [34], [18], [35]. As a result, such models have been used to develop advanced smartphone personal assistant applications that require robust and seamless user experiences [36]. For example, the work in [37] enhances keyphrase extraction by using deep contextual language models to generate rich embedding for each token, which are then processed by a BiLSTM-CRF architecture to capture sequential dependencies and identify keyphrases. This approach leverages domain-specific models to improve performance by providing detailed contextual representations. However, it faces challenges including the dependency on high-quality embedding, high computational and memory demands, and potential issues with generalizing across different domains. Such challenges can affect the overall efficiency and applicability of the model in diverse contexts.

In this thesis, we propose a method that utilizes RNN and LSTM models designed

for sequential or time-series data. To effectively handle massive data quantities, our approach relaxes the frequency of updating converged parameters, updating them once per round or multiple times per rounds with model freezing. By observing the variation in convergence time across model parameters, we reduce the need for repeated local and global updates if a converged parameter persists until the final model convergence. This leads to less computational and memory intensive local training on client devices, addressing the challenges of handling large datasets and ensuring efficient, context-aware personalized services.

2.3 Federated Learning

In the centralized learning approach, a centralized collection of sensitive local data from millions of individuals is prone to privacy risks [23], [38], [39], [40]. Figure 2.2 demonstrates the risk of the centralized approach used by many application providers. The figure illustrated two privacy breach situations. In the first situation, the phone owner shares his private conversation with the central sever. In the second situation, the phone owner encounters online ads as a result of processing his private conversation. This risk can be addressed using the decentralized learning approach, e.g., Federated Learning [4], [23], [15], [40].

In FL, each client locally trains his model. Therefore, data are not shared or stored on a central server. Instead, FL enables distributed training of a global model by users using data locally on user devices [27]. This simply reduces transmission cost and ensures data privacy. As depicted in Figure 1.1, the learning process is often carried

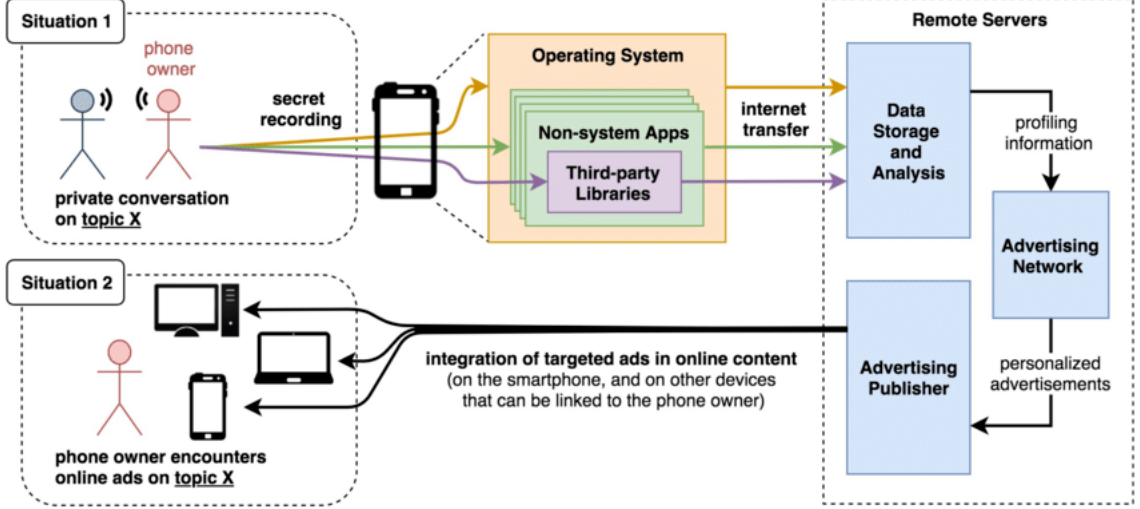


Figure 2.2: *Centralized Learning Architecture [2].*

out by a distributed group of participating clients controlled by a server. Rather than transferring local data to the server, clients compute changes to the shared model managed by the server, and only this update is delivered. Figure 2.3 illustrates the typical FL architecture, where the server aggregates the update of the participating clients and produces a new global model using an averaging operation [21], which is known as Federated Averaging (FedAvg).

2.3.1 State-of-the-Art Algorithms in FL

The Federated Averaging algorithm (FedAvg) works as follows. In the training round t , a global model w_t is sent to a subset K of client devices. At $t = 0$, the global model, which is pre-trained on proxy data or randomly initialized, is considered as the starting point of client devices. Each of the clients participating in a given round has a local dataset consisting of n_k examples, where k is an index of participating clients and n_k varies from device to device. In the case of the personal assistant application, n_k is

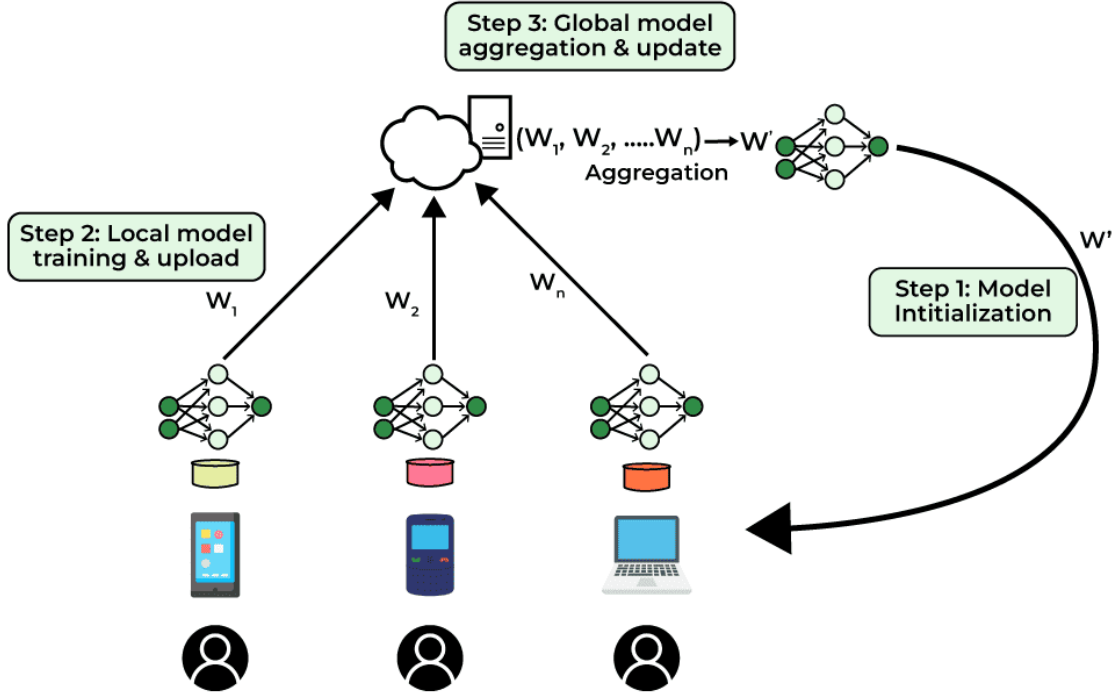


Figure 2.3: *Federated Averaging Architecture* [3].

related to the user's speech inputs and text entries. Each client executes one or more steps of stochastic gradient descent (SGD) to calculate the average gradient g_k in its local data with the current model w_t . Using a learning rate η , the local client update w_{t+1}^k is given by:

$$w_t - g_k \rightarrow w_{t+1}^k$$

The server aggregates the client models to generate an updated global model, represented as w_{t+1} :

$$\sum_{k=1}^K \left(\frac{n_k}{N} w_{t+1}^k \right) \rightarrow w_{t+1}$$

Algorithm 1: Federated Learning Algorithm

Input: Number of training rounds t , learning rate η , number of clients K , fraction of participating clients C , local minibatch size B , number of local epochs E

Output: Updated weights w

// Server executes:

```
1 initialize  $w_0$ ;  
2 for each round  $t = 1, 2, \dots$  do  
3    $m \leftarrow \max(C \cdot K, 1)$ ;  
4    $S_t \leftarrow$  (random set of  $m$  clients);  
5   for each client  $k \in S_t$  in parallel do  
6     ClientUpdate( $k, w_t$ );  
7   end  
   // ClientUpdate( $k, w$ ): // Run on client  $k$   
8   Function ClientUpdate( $k, w$ ):  
9      $B \leftarrow$  (split  $P_k$  into batches of size  $B$ );  
10    for each local epoch  $i$  from 1 to  $E$  do  
11      for each batch  $b \in B$  do  
12         $w \leftarrow w - \eta \nabla \ell(w; b)$ ;  
13      end  
14    end  
15    return  $w$  to server;  
16 end
```

were $N = \sum_k^K n_k$. Algorithm 1 lists the pseudocode of the federated average.

Similarly to FedAvg, FedProx is a distributed algorithm in which the deviation of the local updates from the most recent global model is controlled by introducing a round-varying proximal term. A participating worker uses a proximal update that requires the resolution of a minimization problem [41], [42], [43]. This proximal term mitigates the adverse effects of heterogeneity by stabilizing the training process. It does this by penalizing updates that deviate significantly from the global model, thus providing a more controlled integration of client updates even when only partial infor-

mation is available. The approach theoretically ensures better convergence properties and, empirically, has been shown to improve the stability and accuracy of federated learning, particularly in settings with high heterogeneity, but it has a drawback of a potential computational overhead associated with the proximal term. The additional penalty can increase the complexity of the optimization, potentially requiring more iterations or computational resources to achieve convergence. This overhead can impact the efficiency of the federated learning process, especially in scenarios with limited computational capacity or where the model needs to adapt quickly to new data.

Other recent work includes communication-mediated federated learning (CMFL) [44] and federated learning through clustering optimization (FedCO) [45], which propose clustering-based FL solutions to alleviate high communication costs. This is achieved by grouping workers with similar model parameters and selecting one representative per group. This technique reduces communication overhead. Experimental results demonstrate its superiority over standard FL methods in reducing costs while maintaining accuracy.

More recently, communication-efficient federated learning with adaptive compression in dynamic bandwidth (AdapComFL) [19] addressed the challenge of a large communication overhead in federated learning by introducing adaptive compression based on dynamic bandwidth prediction. Clients perform bandwidth awareness and prediction, adapting compression levels via an improved sketch mechanism. Experimental results based on real bandwidth data and benchmark datasets show that AdapComFL achieves efficient communication and competitive accuracy compared to existing algorithms. Adaptive Parameter Freezing (APF) [26] reduces communica-

tion in federated learning by adaptively freezing stable parameters and periodically checking their need for updates, significantly lowering the transmission volume without compromising convergence. However, it may lead to sub-optimal training due to premature freezing, adds complexity in managing freezing periods, and can destabilize training in large models if stability metrics are inaccurate. We compare our proposed framework with FedAvg [21] and FedProx [4], which focus on minimizing the communication overhead.

2.3.2 Security and Privacy Challenges in FL

Keeping raw data local on each device in federated setups is often required due to privacy concerns [4], [46], [47]. However, numerous literature [48], [49], [50] discussed emerging challenges for privacy-preserving algorithms in federated learning scenarios. Although FL offers several definitions of privacy, they generally fall into two groups: global privacy [51] and local privacy [52], [53]. For global privacy, the updated model generated at each round must be secret to the central server and to all other untrusted third parties, while local privacy further demands that the changes be private to the server. These privacy concerns also exist in the model training and prediction stage [18]. These stages are compounded by the distribution of private data, the number of participants, and the communication methods between participating clients and the server.

Ensuring robust privacy protection involves developing protocols that address these multifaceted concerns, especially in federated learning environments where data remain decentralized. However, organizations are not permitted to collect data from end

devices or local clients to a centralized server to fine-tune pre-trained language models (PLM) due to growing privacy concerns and the adoption of data protection laws [54]. The authors of [54] discuss the need to investigate FL for fine-tuning PLMs without transmitting data to a central server, highlighting the communication overhead and adaptation costs associated with large PLMs.

Recent privacy-preserving techniques such as differential privacy [51] and secure aggregation [55] have attracted a lot of research in FL. Bonawitz et al. [56] introduce a secure aggregation mechanism to protect individual model updates. The precise aggregated results at each round may still be seen by the central server, even if it cannot view any local changes. Although secure aggregation is a lossless technique that may maintain the original accuracy with a very high level of privacy assurance, the resulting approach comes with a large additional cost in transmission. Some studies, for example, [57], provide global differential privacy and apply differential privacy to federated learning; these systems have several hyperparameters that impact accuracy and communication and need to be selected with care. In the context of personal assistant services, recent works like [58], [59]. Wang et al. [59] proposed a DNN-based recommender system, PrivRec, which leverages FL to improve prediction accuracy while preserving user privacy by keeping data on personal devices. To address data heterogeneity, it incorporates a meta-learning approach for quick on-device personalization. The enhanced model, DP-PrivRec, adds differential privacy to protect against malicious participants, ensuring that individual user data cannot be identified. This method is relevant to developing a privacy-preserving personal assistant for keyphrase extraction, as it can optimize the assistant’s ability to handle diverse and sensitive

data while ensuring user privacy through FL and differential privacy techniques.

Our proposed framework, CEFEL, aims to preserve user data privacy by leveraging efficient model parameter freezing methods. The proposed approach contributes to the defense against privacy attacks while minimizing resource consumption as the entire model dimensional space is transmitted intermittently.

2.3.3 Communication-Efficient FL

A significant performance limitation for FL is the communication overhead between clients and the central server [2], [21], [23], [60]. In essence, FL requires edge clients to communicate with the server via the Internet. This makes it possible for programs like Google Gboard to offer user support globally [14]. However, the average bandwidth of the existing Internet infrastructure is limited at the edge [19]. As such, it is crucial to create communication efficient techniques that iteratively deliver brief messages or updates to the models as part of the training process [44], [54], rather than sending the entire data set over the network, to fit a model to the data supplied by the devices. Two important factors to take into account to further reduce communication in such a situation are: 1) reducing the total number of communication rounds as in [22], [24], [45], and 2) reducing the size of the messages transmitted in each round as in [44], [61]. At the same time, it is vital to create techniques that are communication-efficient without unduly sacrificing accuracy, in addition to strong privacy assurances.

In the context of federated networks, communication is a critical bottleneck as the participating clients report their model updates to the server, particularly in fully heterogeneous large-scale networks with limited resources, as seen in Figure 2.4. Fed-

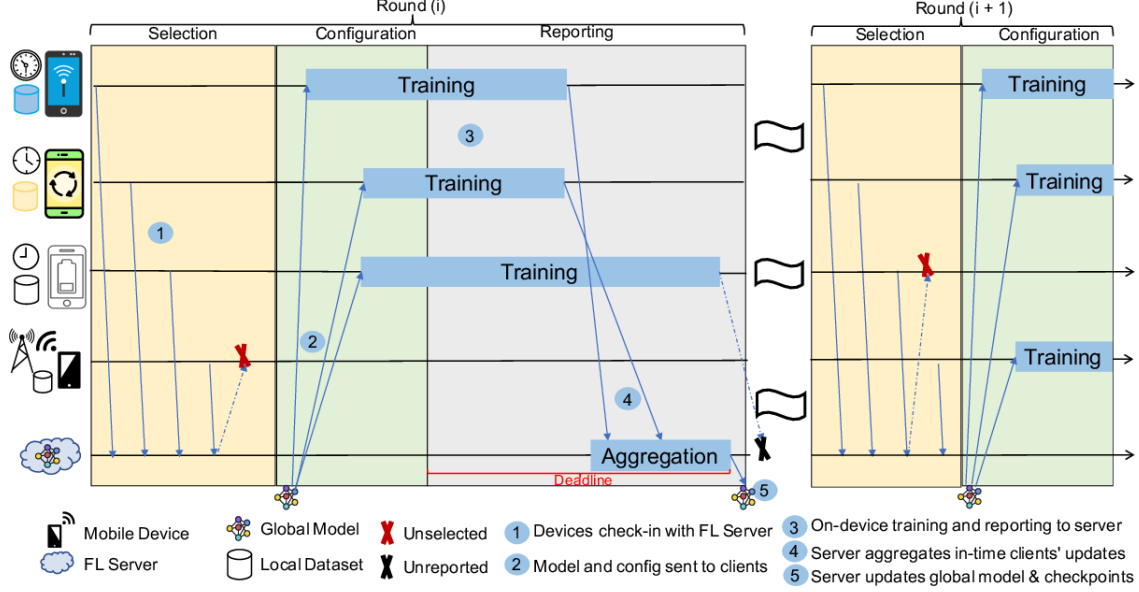


Figure 2.4: Communication Bottleneck in Federated Learning [4].

erated learning approaches in [24], [62], [63], [64] are made to manage several devices collecting data while a central server manages the global learning goal throughout the network. In federated optimization, it is standard practice to employ a local objective function based on device data as a twin for the global objective function on each device in order to minimize communication. A portion of the devices are chosen at each outer iteration, and the local objective functions on each of the chosen devices are optimized using local solvers. The central server receives the changes from the local models of the devices, combines them using an averaging algorithm, and modifies the global model appropriately [65].

Numerous studies have sought to reduce communication costs in distributed model training, proposing strategies such as quantization [25] and increasing local computation [21] to minimize data transmission. In the context of system heterogeneity, the authors of [4], [65], [66] demonstrated empirical success in using FedAvg in heteroge-

neous settings, but failed to completely address the underlying challenges associated with heterogeneity. FedAvg does not allow participating devices to perform variable amounts of local work based on their underlying systems constraints; instead it is common to simply drop devices that do not compute E epochs within a specified time window [48]. It typically operates synchronously, requiring all participating clients to finish their local updates before aggregating the results. This can lead to communication delays if even one client is slow or has intermittent connectivity.

Recent works have introduced new techniques in reducing communication costs in FL, e.g. [19], [26], [67], [68], [69], [70], [71], which suggest freezing stable model parameters to mitigate communication overhead. The work of Chen et al. [72] Adaptive Parameter Freezing (APF), proposes a novel scheme to reduce communication overhead in federated learning by excluding stable parameters from global synchronization at an early stage. By tentatively adjusting freezing periods, APF effectively mitigates the divergence of local parameters between clients, resulting in a significant reduction in data transfer without compromising model accuracy. APF demonstrates a reduction in 60% in data transfer. However, with APF, Parameters that are prematurely frozen might miss important updates if their stability is incorrectly assessed, potentially leading to sub-optimal training and dynamically adjusting freezing periods for each parameter adds complexity to the training process, requiring careful management and tuning. Also, APF is particularly effective for over-parameterized models, and its benefits might be less pronounced in models that are not over-parameterized and in models with parameters that stabilize late or not at all. In FL [73], Automatic Layer Freezing (ALF) significantly reduces communication costs by progressively ex-

cluding certain ConvNet layers from training, achieving up to 83.91% reduction in data exchange with minimal impact on model accuracy. ALF seamlessly integrates with FL strategies like FedAvg and FedProx and adapts dynamically to various models and datasets. A drawback of ALF is its reliance on predetermined freezing heuristics, which may not optimally adapt to different architectures. Using LSTM networks as demonstrated in [74], [75] can address this by dynamically adjusting the freezing criteria based on the real-time sequence performance, enhancing the effectiveness of ALF in managing communication and improving model convergence.

Our research complements the freezing techniques introduced by the works of [26], [72] by emphasizing on a similar drawback, where the performance gain from parameter freezing may be limited in models with parameters that stabilize late or not at all. Our methodology effectively tackles both the issue of late convergence of model’s parameterized layer and the challenges of partial and permanent freezing, providing a comprehensive solution to cutting down communication costs in federated learning, particularly in a heterogeneous environment.

2.4 Federated Learning for Personal Assistant Applications

In the context of personalized services, the following surveys [47], [76], [46] highlight the need for personalized federated learning (PFL) to address statistical heterogeneity, providing a taxonomy of strategies such as data-based, model-based, architecture-based, and similarity-based approaches. By focusing on PFL, it fills a gap in the

literature and suggests methods to improve global model generalization and local adaptation. For developing a personal assistant that predicts and extracts keyphrases in an FL setting, these insights can inform strategies to handle data diversity and personalization, ensuring effective and accurate keyphrase extraction tailored to individual user data while maintaining privacy.

For personal assistant applications that use large-scale language models as used in [54], [72], various distributed optimization techniques, including those developed by [44], [45], [4] to establish the suitability of such large-scale models in a FL environment, the authors of [77] demonstrated that training language models with a large vocabulary can be accomplished while maintaining accuracy and privacy via FL. However, most current NLP and FL works [78], [79], [76], [47] are based on PLMs, such as [54], [80], [81]. Our research direction investigates large-scale language models in a communication-efficient federated learning environment without requiring prior training.

In deploying FL for model training, the authors of [21] increased the amount of local computing while decreasing the frequency of general communication to achieve the desired level of communication efficiency. This reduced the number of local calculations needed by each client for each epoch and allowed for more local updates each communication round. These techniques necessitate strong processing capacity and because local training is prioritized, there could be variations in the global model’s performance between clients. In addition, the survey conducted by [27] concluded that the emergence of massive mobile phone data, along with advanced computing methods, has significantly advanced this domain. Their survey shows that mobile phones, as

personal devices, allow for the inference of real social networks from call-detail records (CDRs). Unlike fixed phones, which reflect a superposition of several social actors, mobile phones provide detailed communication patterns and positioning data to track users' movements. These advances present challenges in handling large amounts of data, inferring social networks, capturing temporary patterns in communication, and tracking movements, which require sophisticated machine learning models.

CHAPTER 3

CEFEL FRAMEWORK

In this section, we introduce the proposed Communication-Efficient Federated Learning (CEFEL) framework. First, we present the design of the framework that involves developing the personal assistant application, developing the machine learning POS tagging model, and implementing the federated learning environment. Then, we present the CEFEL strategy to minimize communication rounds while maintaining the validity of the model’s convergence. Finally, we provide the design and operation of our proposed enhanced federated learning freezing techniques that guarantee communication efficiency using our CEFEL framework. Figure 3.1 depicts the overall proposed framework.

3.0.1 Application Development

In this section, we discuss the development of the personal assistant application. We used the Flutter software platform in the design of the user interface of the application as shown in Figure 3.2. Similar to the work in [1], [17] our developed smartphone

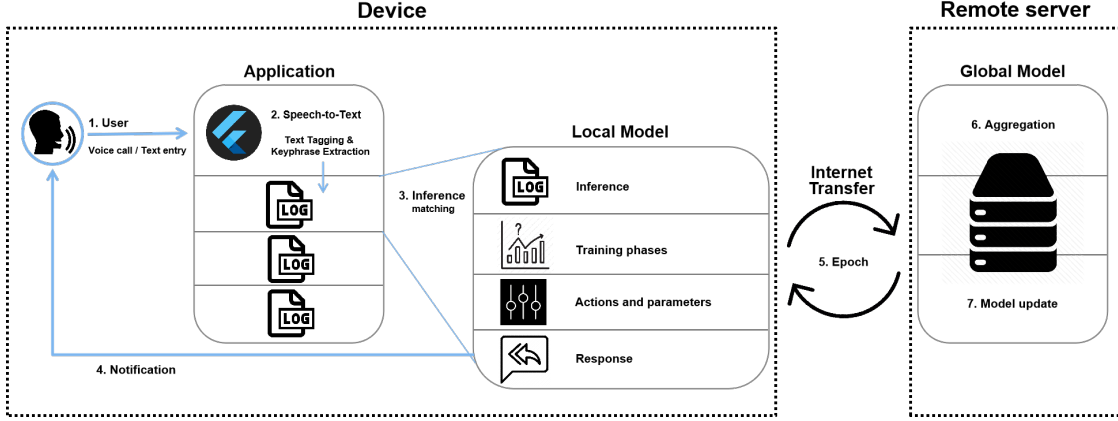


Figure 3.1: Proposed Intelligent Personal Assistant Utilizing CEFEL Framework.

personal assistant application for keyphrase extraction consist of the following components: 1) a speech-to-text module that translates spoken words into written language, 2) a semantic context parser that segments the text document into individual sentences and tokenizes each sentence, 3) a POS tagger that uses machine learning algorithms to predict the POS tags of each token , 4) a keyphrase vectorizer to extract keyphrases based on selected part-of-speech pattern, and 5) a Dialog Manager that matches the predicted keyphrases to some rule-based logic to trigger the desired action on the system. The modules are implemented using a publicly available library and an in-built feature of STT API that include various ML and NLP algorithms to achieve the conversion of speech to text, even in noisy environments. We used a keyphrase vectorizer, an open source Python module [1], to extract keyphrases based on accurate predictions of POS tags and a match with the selected POS pattern. For the in-built Flutter STT library, the target use cases for this library are commands and short phrases, not continuous spoken conversion or always listening. The output of this STT API is expected to be the input instance of the trained model for the part-of-speech tagging and keyphrase extraction module, both of which facilitate the

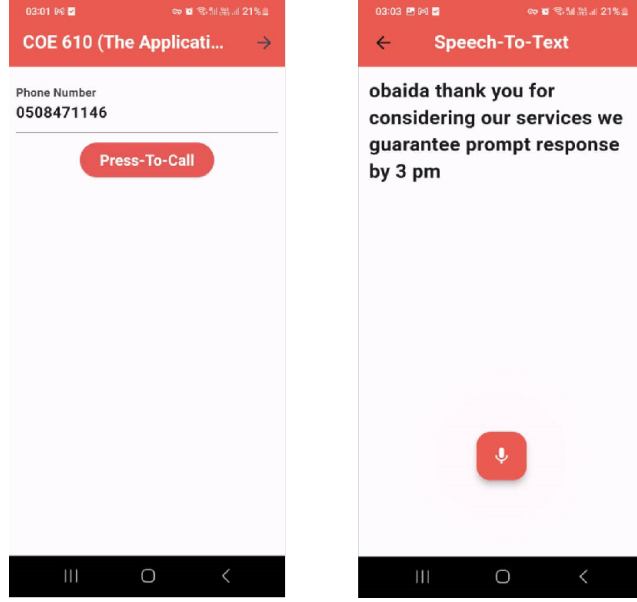


Figure 3.2: Speech-to-Text Application.

triggering of an action.

3.0.2 Communication-Efficient FL via Model Parameter Freezing

Our FL algorithms proposed in an efficient communication model under the CEFEL framework are called **FedFreeze** and **FedFreeze+**. These algorithms address the critical bottleneck of communication in FL by introducing comprehensive techniques to reduce communication overhead. The proposed approaches are inspired by the concept of freezing convergent model parameters [26], [72], [73]. In particular, our method optimizes the synchronization process between the participating clients and the server. The essence of the proposed algorithms is the efficient use of the convergence curve pattern of each trainable layer in the model to appropriately freeze (F) or unfreeze (UF) converged parameters, as shown in Figure 3.3. As illustrated in the

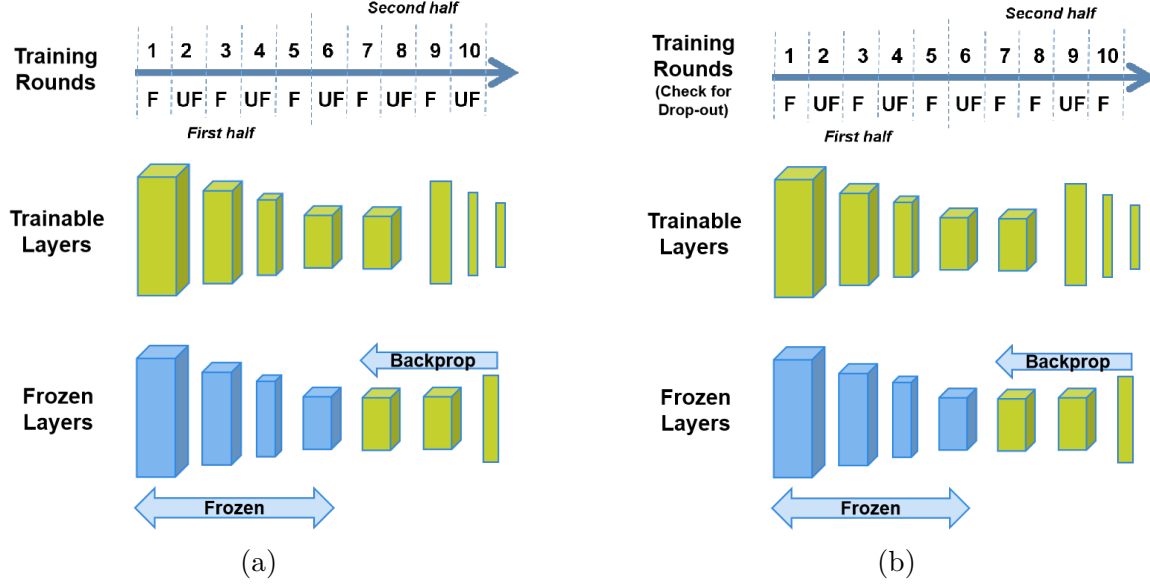


Figure 3.3: Freezing Demonstration of 8 Trainable Layers for 10 Training Rounds with (a) FedFreeze Architecture. (b) FedFreeze+ Architecture.

figure, the first approach (FedFreeze) alternates parameter freezing across training rounds as shown in Figure 3.3a, while the second approach (FedFreeze+) performs incremental parameter freezing to lower the transmission volume during the FL process as shown in Figure 3.3b. Although the freezing technique can be an effective solution to reduce communication overhead, previous studies identified several challenges and their deficiencies in freezing selected layers or individual parameters of a model [26], [72], [63], [73]. Following our proposed approach, we have to address the following challenges associated with the freezing technique: 1) temporary convergence of parameters, 2) limited resources for continuous convergence checks of individual parameters or layers, and 3) late convergence of parameters.

Partial Freezing Problem. An obvious option would be to merely synchronize the remaining parts of the model to the central server, leaving stabled parameters out of the synchronization process (while updating them locally) in order to take advantage

of the stabilized parameters for communication compression. The authors of [26] discovered that there may be a significant loss in accuracy when using such a partial synchronization technique, since each client participates in the FL environment, with its personalized local data (i.e., non-IID) [19]. As a result, it is implied that there are generally differences between the local loss function $F(w)$ and the related local optima w on various clients [26]. This is supported by recent comparable research work, such as [82], [69], [26], [19], which illustrated and established that, compared to full-model synchronization, the accuracy of the model with such a partial synchronization technique results in a loss of accuracy greater than 10%.

Permanent Freezing Problem. Permanent freezing of parameters can result in suboptimal accuracy compared to full-model synchronization, as it prevents the parameters from reaching their true optimum. This issue arises because distinguishing temporarily stabilized parameters from truly converged ones is inherently difficult based on local observations. Consequently, premature freezing of these parameters can limit the benefits of the freezing technique, leading to late convergence and compromised model accuracy [26].

Late Convergence Problem. The two problems explained above and captured by many existing research still fall short of the late convergence problem. However, our research identifies this as a significant drawback in existing approaches such as in [72], [26], [67]. The performance gain from parameter freezing is minimal if the parameters intended to be frozen converge late during the training phase or not at all. This late convergence issue not only limits the effectiveness of freezing, but also increases the computational power and time required, as more iterations are needed to

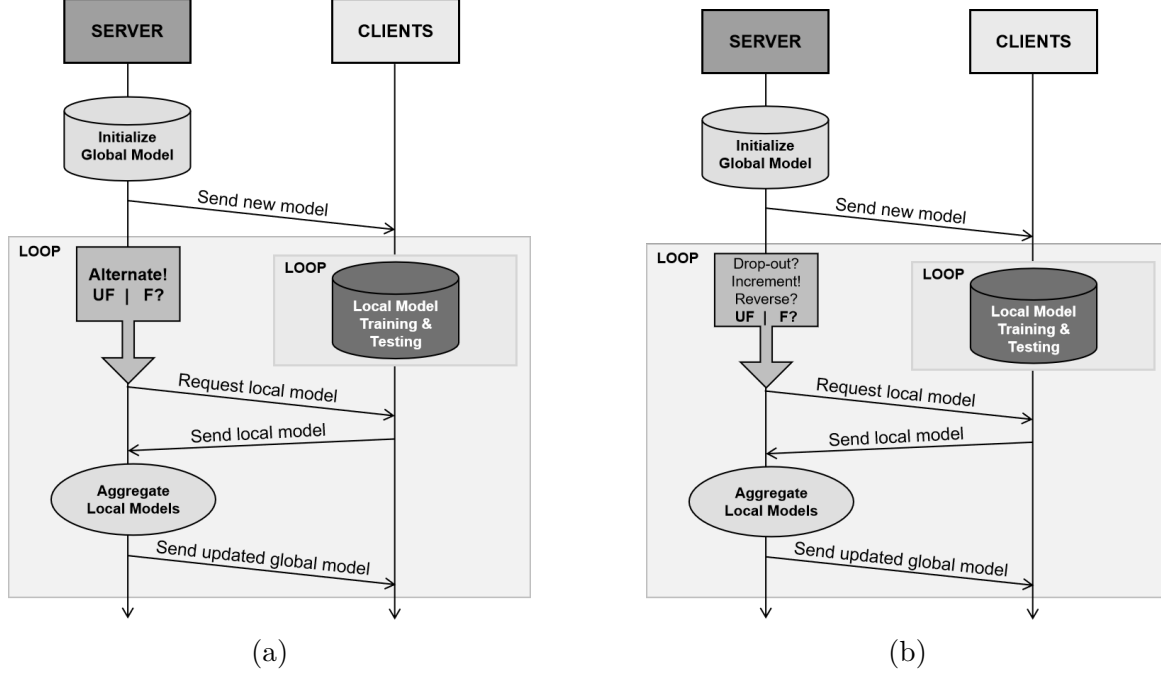


Figure 3.4: CEFEL Workflow (a) FedFreeze. (b) FedFreeze+.

achieve optimal performance. Our methodology addresses both the initially outlined problems and the late convergence issue, thereby enhancing overall model efficiency and reducing both communication and computational costs.

3.1 Freezing Techniques

In this section, we discuss the enhanced mechanisms of alternating parameter freezing (FedFreeze) and incremental parameter freezing (FedFreeze+) to lower the transmission volume during FL.

3.1.1 FedFreeze

FedFreeze is a federated learning algorithm designed to optimize communication efficiency and model convergence by implementing the freezing of layer-wise parameters

within the CEFEL framework. This algorithm operates by alternating the freezing of the layer parameters in a structured manner, as illustrated in Figure 3.4a. As a result, we address the issues of late convergence and increased computation in resources-constrained edge devices as observed in methods introduced in recent works such as those of Chen et al. [26], [72] and Pfeiffer et al. [69]. Unlike these recent approaches, which may suffer from delayed convergence and limited gains due to continuous observation before freezing, FedFreeze introduces a proactive mechanism to enhance training efficiency and reduce communication overhead.

In particular, FedFreeze employs a central server that plays an essential role in orchestrating the training process and managing the freezing mode, as illustrated in Figure 3.4a. At the beginning of each training round, the server distributes the global model to all participating clients, accompanied by a bit of signal. This signal indicates whether the upcoming local training phase should operate in freezing mode, in unfreezing mode, or follow the standard FedAvg approach of full model updates. If the signal indicates freezing mode, clients are instructed to freeze the parameters of a specified layer, thus excluding them from further updates during local training. In contrast, if the unfreezing mode is signaled, clients should unfreeze previously frozen layers and include them in the training updates. In the absence of any signal bit, the clients default to the standard FedAvg approach, updating all the parameters of the model. The impact on the signal bit in terms of volume is insignificant to introduce overhead and more so will be compensated by the gains of continuous freezing. In the following sections, we discuss in detail the FedFreeze procedures.

Initialization and Model Distribution

The central server initializes the global model and sends it to the clients, along with the bit of the signal. The signal bit dictates the mode: freezing, unfreezing, or full-model update.

Local Training

There are three modes of backpropagation during local training: Freezing mode, Unfreezing mode, and Standard mode.

1. **Freezing Mode:** Clients freeze the parameters of the designated layer as per the server’s instructions. During local training, the frozen parameters remain unchanged, and no updates are performed on them. This effectively reduces the computational burden and the volume of data to be communicated back to the server.
2. **Unfreezing Mode:** Clients unfreeze the previously frozen layers and reintegrate them into the model for training. Before local training, the global model is reconstructed to include these layers. Backpropagation is performed on the entire model, updating all parameters including those previously frozen.
3. **Standard FedAvg Mode:** Clients perform a full model update, training all layers without any freezing. This occurs when no specific signal bit is provided by the server as illustrated in Algorithm 2, ensuring flexibility to adapt to different training phases.

Parameter Aggregation and Update

After local training, the clients send their updated parameters back to the central server. In freezing mode, only the updates for the non-frozen layers are transmitted, while the frozen layers remain static. The server aggregates these partial updates and integrates them into the global model. In unfreezing mode, updates include previously frozen parameters, allowing the server to fully reconstruct the global model before integrating the new updates. In standard FedAvg mode, all parameters are updated and the server performs a typical aggregation of the entire model. With the server dynamically alternating the signal bit based on the hardcoded system configuration, it optimizes and balances between communication efficiency and training effectiveness.

The FedFreeze algorithm significantly improves federated learning by reducing communication overhead and improving training efficiency. The proactive and alternating freezing schedule allows clients to benefit from model parameters that converge early without impacting the model accuracy and reduces the volume of data exchanged between clients and the server by up to 35.30% in experimental evaluations. Furthermore, by integrating freezing signals and mode-specific backpropagation, FedFreeze offers a flexible yet systematic approach to managing training dynamics, ensuring better model performance in heterogeneous and resource-constrained environments.

3.1.2 FedFreeze+

FedFreeze+ enhances the FedFreeze algorithm by incorporating "incremental freezing", which is adaptive and responsive to client dropout rates, to optimize both com-

munication efficiency and model convergence during federated learning. In particular, FedFreeze+ refines the parameter freezing strategy by leveraging observed convergence patterns. As depicted in Figure 3.4b, in the first half of the training period, FedFreeze+ mimics the alternating freezing approach of FedFreeze. However, it dynamically adjusts the number of freezing rounds based on the dropout rates of the participating clients. When client dropout is minimal, the algorithm increases the duration of freezing rounds, capitalizing on stable participation to minimize communication overhead. In contrast, if dropout rates increase significantly, it reduces the duration of freezing proportionately, ensuring that the model maintains adequate training opportunities to adapt to the fluctuating client base. Using dropout as a hyperparameter, this approach allows FedFreeze+ to flexibly respond to variations in client availability, striking a balance between communication efficiency and the need for model updates.

In the second half of the training period, FedFreeze+ introduces the concept of "incremental freezing", where the parameter freezing is applied multiple times after each round. This method builds on previously established convergence insights, enabling layers that have shown early convergence to remain frozen for longer periods, while layers that have not converged undergo more frequent evaluations for freezing. This incremental strategy effectively reduces the communication and computation costs by limiting the need to transmit converged layer updates repeatedly, while allowing layers that have not converged to adapt and converge adequately. FedFreeze+ demonstrates an optimal reduction in data exchange volumes and computational efforts, with experimental results indicating up to 40.95% in communication cost savings without sig-

nificant detriment to model accuracy. The algorithm also supports controlled depreciation in accuracy, making acceptable trade-offs to enhance efficiency while preserving performance. This nuanced approach ensures that FedFreeze+ achieves a practical balance, allowing federated learning systems to operate efficiently even in the face of client dropout and varying training dynamics. The rate of "dropout" among clients dynamically adjusts the freezing frequency of model parameters in FedFreeze+. In the following sections, we discuss in detail the FedFreeze+ procedures using the following notations.

- R_t : Number of rounds for freezing at time t
- R_{base} : Base number of rounds for freezing
- D_t : Dropout rate at time t (proportion of clients dropping out)
- ΔR_t : Incremental change in freezing rounds at time t
- α : Proportionality constant for incrementing freezing rounds based on dropout
- β : Proportionality constant for decrementing freezing rounds based on dropout
- γ : Threshold for significant dropout change

Initialization

The initialization procedure starts with a base number of rounds R_{base} to freeze.

$$R_0 = R_{\text{base}}$$

Incrementing Freezing Frequency

When dropout is below the threshold ($D_t \leq \gamma$), increase the freezing frequency:

$$\Delta R_t = \alpha \cdot (1 - D_t)$$

$$R_{t+1} = R_t + \Delta R_t$$

Decrementing Freezing Frequency

When dropout exceeds the threshold ($D_t > \gamma$), decrease the freezing frequency:

$$\Delta R_t = \beta \cdot D_t$$

$$R_{t+1} = \max(R_{\text{base}}, R_t - \Delta R_t)$$

For the threshold value γ and the increment factors α and β , fixed values were used for the purpose of the experiment. The proportionality constants are set as $\alpha = 2$ and $\beta = 1$, with a threshold value of $\gamma = 0.2$. The higher values of α and β will result in more aggressive adjustments in the freezing frequency, while the lower values will lead to more conservative changes. The server plays a crucial role in FedFreeze+, as it monitors the rates of client dropout, computes the model formulation as illustrated above, and adjusts the freezing strategies accordingly as described in Figure 3.4b.

3.2 CEFEL Framework

In this section, we discuss in detail the proposed CEFEL framework, which uses FedFreeze and FedFreeze+ to lower the transmission volume during FL. We discuss the framework in the context of a personal assistant application that extracts keyphrases from the user’s voice input and triggers an action based on the rule-based logic of the system as shown in Figure 3.1.

Algorithm 2 lists the complete training pseudocode of the model updates on the client and server side, consisting of FedFreeze, FedFreeze+ and FedAvg as the default approach. The algorithm outlines a federated learning framework in which a central server coordinates multiple clients to train a machine learning model. The server initializes the model parameters and, in each round, selects a fraction of clients to participate. These clients perform local updates on the model parameters in parallel. The server employs different parameter freezing strategies: the alternate approach freezes parameters once per round, and the incremental approach freezes parameters multiple times per round based on a threshold as training progresses. If no freezing is specified, standard federated averaging (FedAvg) is performed. The ClientUpdate function, executed by each client, involves splitting the client’s data into batches, performing local training over several epochs, and updating the model parameters using gradient descent. This process repeats for a specified number of rounds, resulting in collaboratively trained model parameters.

Algorithm 2: CEFEL

Input: Number of training rounds T , learning rate η , number of clients K , fraction of participating clients C , local minibatch size B , number of local epochs E , base number of freezing rounds R_{base} , dropout rate threshold γ , increment factor α , decrement factor β

Output: Selected Updated Weights w_t

// Server executes:

```
1 initialize  $w_0$ ;
2  $t \leftarrow 1$ ;
3  $R_0 = R_{\text{base}}$ ;
4 while  $t \leq T$  do
5    $m \leftarrow \max(C \cdot K, 1)$ ;
6    $S_t \leftarrow$  (randomly select  $m$  clients);
7   for each client  $k \in S_t$  in parallel do
8      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$ ;
9   end
10  Compute dropout rate  $D_t$ ;
11  if  $D_t \leq \gamma$  then
12     $\Delta R_t = \alpha \cdot (1 - D_t)$ ;
13     $R_{t+1} = R_t + \Delta R_t$ ;
14  else
15     $\Delta R_t = \beta \cdot D_t$ ;
16     $R_{t+1} = \max(R_{\text{base}}, R_t - \Delta R_t)$ ;
17  end
18  if freezing approach is Alternating then
19    Freeze parameters once after each round;
20    Aggregate only unfrozen parameters to update  $w_{t+1}$ ;
21  else
22    if freezing approach is Incremental then
23      if  $t < T/2$  then
24        Freeze parameters once after each round (Alternating);
25      else
26        Freeze parameters multiple times after each round based on
27         $R_{t+1}$ ;
28      end
29      Aggregate only unfrozen parameters to update  $w_{t+1}$ ;
30    else
31      Normal FedAvg operation:  $w_{t+1} \leftarrow \frac{1}{m} \sum_{k \in S_t} w_{t+1}^k$ ;
32    end
33   $t \leftarrow t + 1$ ;
34 end
35 return Selected Updated Weights  $w_t$ 
// ClientUpdate( $k, w$ ): // Run on client  $k$ 
36 Function  $\text{ClientUpdate}(k, w)$ :
37    $B \leftarrow$  (split  $P_k$  into batches of size  $B$ );
38   for each local epoch  $i$  from 1 to  $E$  do
39     for each batch  $b \in B$  do
40        $w \leftarrow w - \eta \Delta \ell(w; b)$ ;
41     end
42   end
43   return  $w$ ;
```

CHAPTER 4

PERFORMANCE EVALUATION

In this section, we present a comprehensive performance evaluation of the CEFEL framework using the proposed algorithms, i.e., FedFreeze and FedFreeze+. First, we discuss the dataset and two data distribution scenarios, i.e., IID and non-IID. Next, we present the experimental setup to evaluate FedFreeze and FedFreeze+. Then, we discuss the evaluation results in terms of communication cost, computation overhead, and accuracy. Finally, we discuss the deployment of our model and its prediction performance. All experiments were conducted using PyTorch and Flower Framework on an NVIDIA RTX A6000 GPU with 48 GB of memory. Throughout this chapter, we denote training rounds as R , the learning rate as lr , the number of clients as C , the number of dropped clients as d , the size of the local mini-batch as B and the local epoch as E .

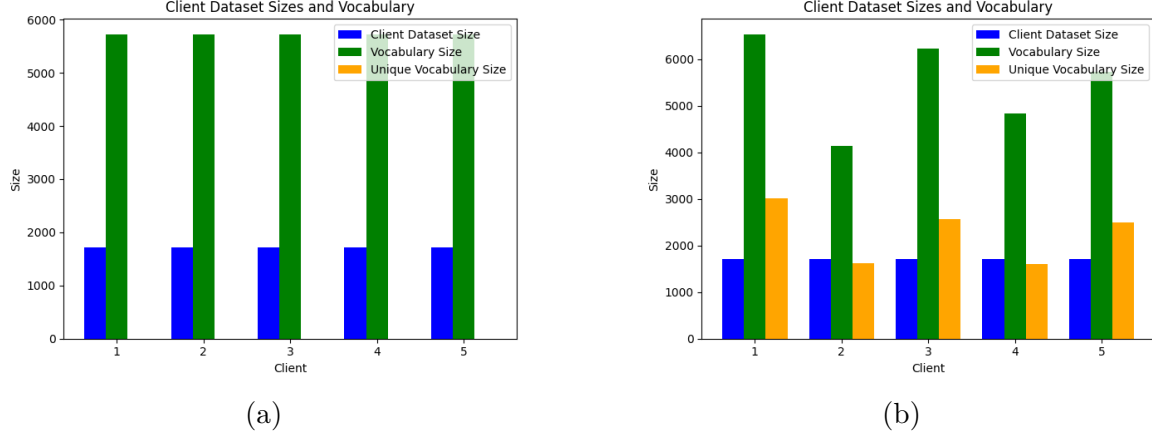


Figure 4.1: Dataset Partitioning across 5 Clients Based on Vocabulary List for (a) IID Distribution. (b) Non-IID Distribution.

4.1 Dataset

In all experiments, we used the Georgetown University Multilayer Corpus (GUM) dataset, which is a rich annotated open-access corpus designed for linguistic research [83]. It consists of 8,548 sentences, a vocabulary size of 16,787, a unique POS tag size of 19 and over 228,000 tokens, i.e., a word, punctuation signs, symbol, and special characters. The dataset includes multiple layers of annotation, including syntax, semantics, and discourse structures, that cover various text genres. GUM dataset is widely used for NLP tasks and linguistic analysis due to its comprehensive and detailed annotations.

IID vs. non-IID datasets. In our experiments, we used the vocabulary list of the corpus to create two scenarios: IID and non-IID. In the IID scenario, the data is distributed uniformly among all participating devices as depicted in Figure 4.1a. In particular, each device has a similar distribution of data with the same vocabulary list. This simplifies the learning process because the data for each device are a repre-

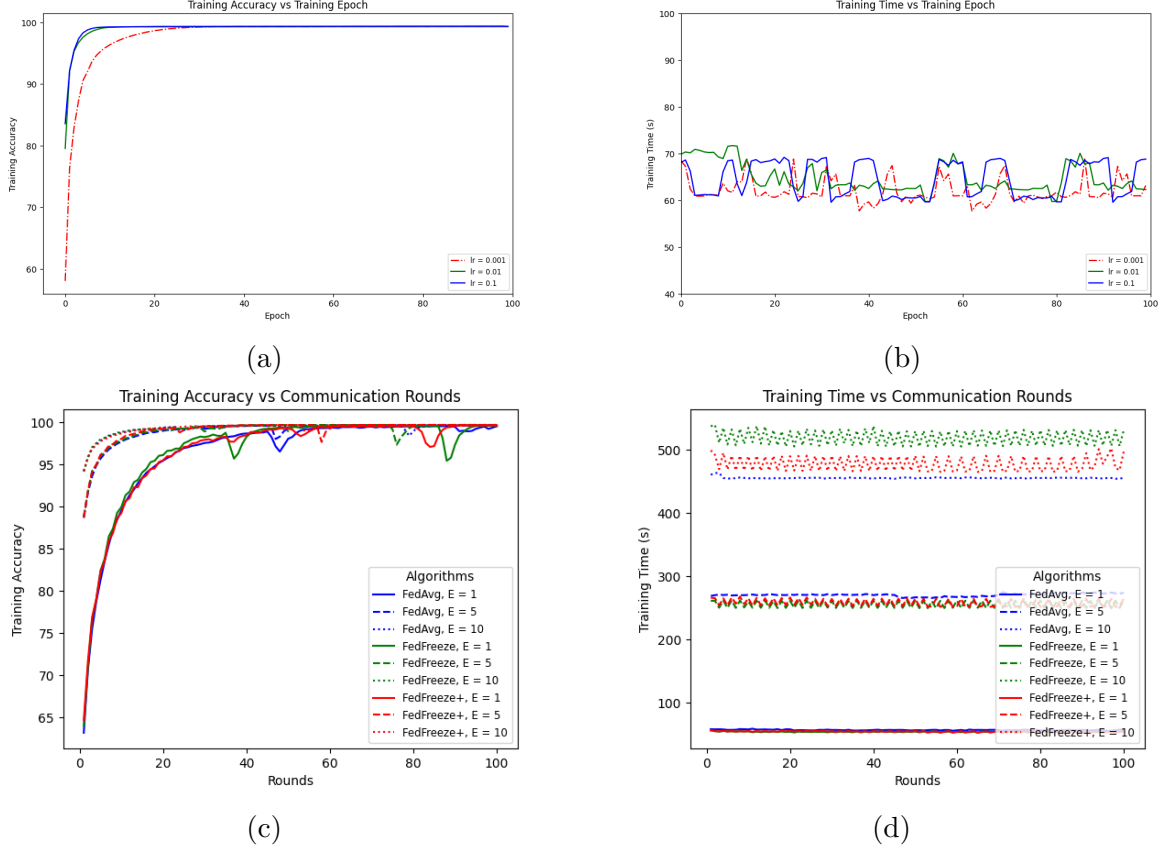


Figure 4.2: Experiments for selecting the best hyperparameters on non-IID Data for (a) accuracy of Best-Model Learning Rate (b) training Time of Best-Model Learning Rate (c) accuracy of Different Number of Local Epoch (d) training Time of Different Number of Local Epoch.

sensation of the overall dataset. On the other hand, in the non-IID scenario, the data is unevenly distributed between devices, where each device has an incomplete representation of the general vocabulary list, but maintains all target classes as illustrated in Figure 4.1b. This data distribution scenario makes it challenging to train a model that performs well across the general vocabulary list and all classes.

4.2 Experimental Setup

4.2.1 Model Architecture

In our experiments, we adopt the DualLSTMTagger model [84], which is a lightweight LSTM network with five recurrent layers: 2 input embedding layers, 2 hidden LSTM layers, and 1 output layer. The dimension of the embedding layer is 300 and 128, while the dimension of the hidden layers is 512 and 128.

4.2.2 Hyperparameters Tuning

In all of our experiments, we run the FL training stage for 100 communication rounds on 10 and 30 clients. We conducted extensive experiments to select the best hyperparameters for our experiments. Figure 4.2 shows various experiments to select the optimal learning rate lr and epochs E . Based on previous work with LSTM-based models, e.g. [24], [70], we have used the range of 0.001 to 0.1 to adjust the learning rate. Figures 4.2a and 4.2b show the training accuracy and the training time compared to the varying learning rate lr (0.1, 0.01, and 0.001) using a batch size B of 100. The figures show that a learning rate of $lr = 0.1$ has the best training accuracy as shown in Figure 4.2a and convergences faster than $lr = 0.01$ and $lr = 0.001$. Furthermore, $lr = 0.1$ generally requires less training time, as illustrated in Figure 4.2b. As a result, we performed all our experiments with a learning rate of $lr = 0.1$. Figures 4.2c and 4.2d show the accuracy and training time of FedAvg against varying local training epochs (1, 5, and 10) as performed in [70]. The figures show that the use of $E = 5$ provides a faster convergence in the training precision than $E = 1$ and

Algorithm	Learning Rate (lr)	Epochs (E)	Rounds	No. of Client (C)
SGD (Centralized)	0.1	100	-	-
FedAvg	0.1	1,5	100	10,30
FedFreeze	0.1	1,5	100	10,30
FedFreeze+	0.1	1,5	100	10,30

Table 4.1: Experiment Hyperparameters.

Layer	(Vocab Size, Dim.)	Number of Parameters	Percentage
Word Embedding	(16788, 300)	5,036,400	55.5%
Char Embedding	(200, 128)	25,600	0.3%
Char LSTM	(128, 128)	65,536	0.7%
LSTM	((300 + 128), 512)	3,932,160	43.4%
Linear	(512, 19)	9,728	0.1%
Total	-	9,069,424	100%

Table 4.2: Breakdown of Model Parameters by Layer.

is slightly slower than the use of $E = 10$. However, $E = 5$ requires less training time compared to $E = 10$. We observed that using $E = 5$ in FedFreeze and FedFreeze + maintains the same precision and training time with FedAvg as shown in Figures 4.2c and 4.2d. Table 4.1 summarizes the values of the hyperparameters used in the rest of the experiments.

4.2.3 Ablation Study

In this experiment, we verify the scalability of FedFreeze and FedFreeze+ by varying the hyperparameters. Furthermore, we conduct an ablation study on the DualLSTM-Tagger model architecture to evaluate the best layer parameters to be frozen and offer an acceptable trade-off in terms of model accuracy, communication costs, and computational overhead. The results are shown in Figure 4.3. The DualLSTMTagger model consists of 5 recurrent layers: *word_embedding*, *char_embedding*, *char_lstm*, *lstm*,

Algorithm	C = 10	R = 100	Cost (s)	Reduction
FedAvg		254.08		-
FedFreeze		210.13		17.30%
FedFreeze+		253.58		0.20%

Algorithm	C = 30	R = 100	Cost (s)	Reduction
FedAvg		270.12		-
FedFreeze		255.37		5.46%
FedFreeze+		257.66		4.61%

Table 4.3: Average Computation Cost Per Round in Seconds (s) Between CEFEL and FedAvg for non-IID Distribution and $E = 5$.

and hidden2tag. Table 4.2 shows the size, number of parameters and percentage of each layer in the model architecture. The word_embedding and the char_embedding layers both converge very early compared to the other layers investigated during the local training phase as seen in Figure 4.3a, while in the federated phase, having both layers or each of the layers frozen maintains a closely similar model accuracy and training time as seen in Figures 4.3b and 4.3c. However, as seen in Figure 4.3d, freezing the word_embedding layer reduce the FL communication cost by 40.95% due to its high parameter percentage of 55.5% of the model as listed in Table 4.2. Hence, further experiments were conducted with only the word_embedding layer frozen without freezing both the word_embedding and the char_embedding layers to avoid the accuracy reduction seen in Figure 4.3b despite a huge communication cost reduction as seen in Figure 4.3d.

4.3 Evaluation Results

We compare FedFreeze and FedFreeze+ with the baseline algorithms, i.e., FedAvg, using three metrics: communication costs, computation overhead, and model accuracy.

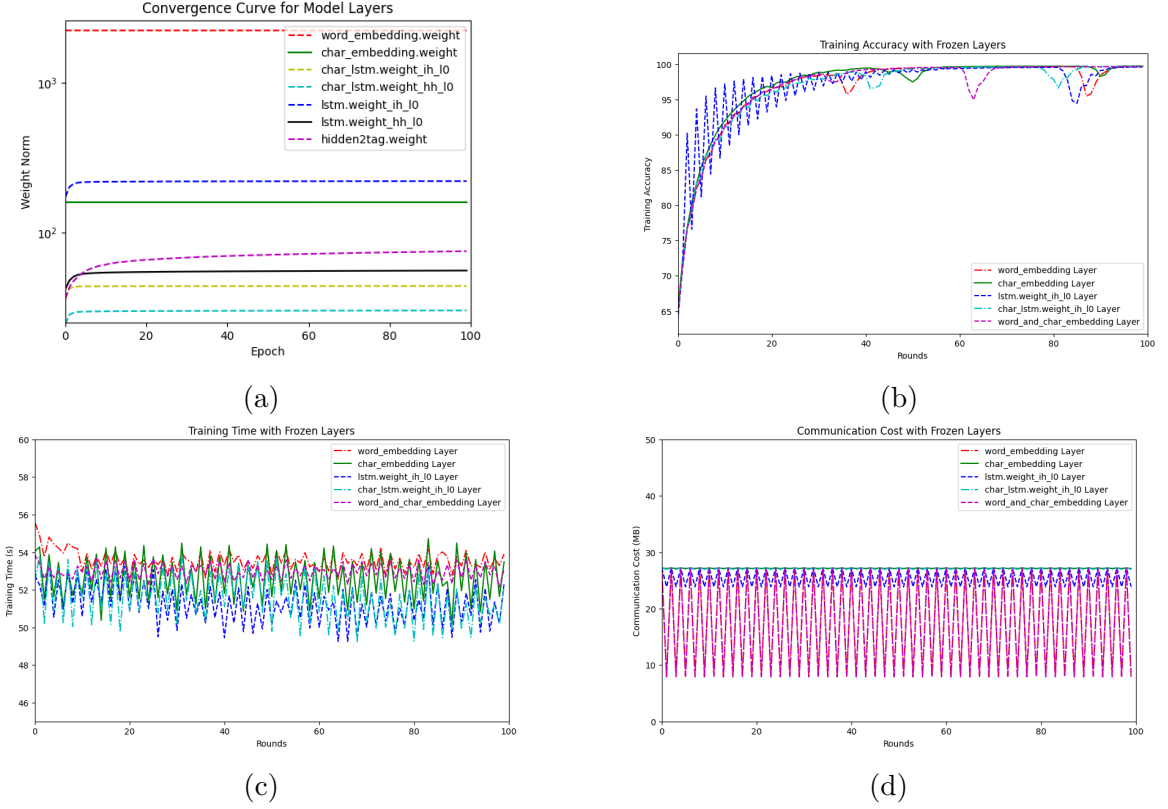


Figure 4.3: Ablation Study for (a) Convergence Curves of Model Layers (b) Accuracy with Frozen Layer (c) Training Time with Frozen Layer (d) Communication Cost with Frozen Layer

4.3.1 Communication Cost

The local models on each client were trained using the SGD optimizer. Figure 4.4a shows FedAvg used as a benchmark to evaluate the improvement in reducing communication costs by many research works in this area, most of which are small-scale classification models. Based on our proposed approach, Figure 4.4b shows our success in reducing communication overhead compared to standard FL algorithms by about 35.30% for FedFreeze and 40.95% for FedFreeze+. The reduction is due to the freezing of the parameters during the training rounds.

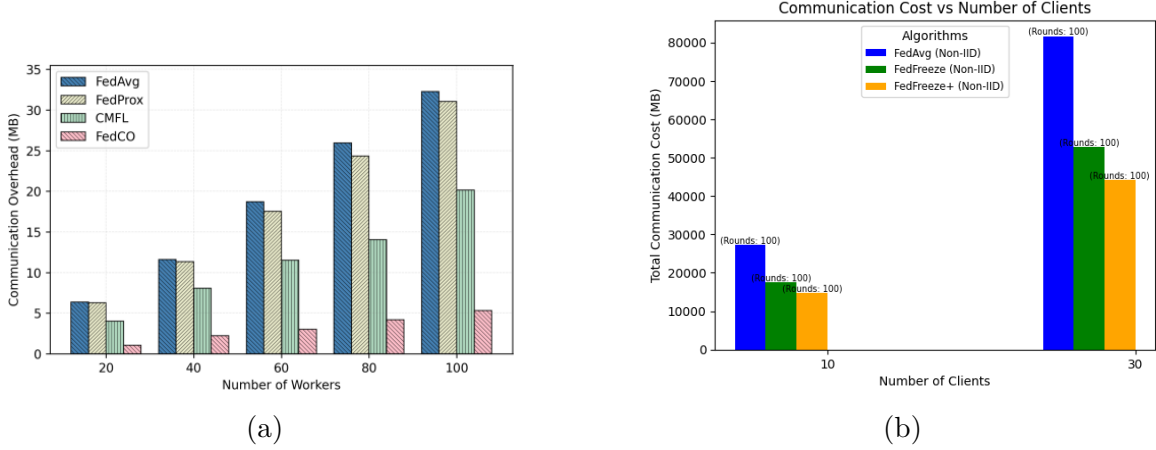


Figure 4.4: Comparison of Communication Overhead Against Number of Clients for (a) Existing Frameworks on 100 Rounds for the CIFAR-10 with non-IID using CNN [5]. (b) GUM Datasets with non-IID on CEFEL Framework for 100 Rounds using RNN.

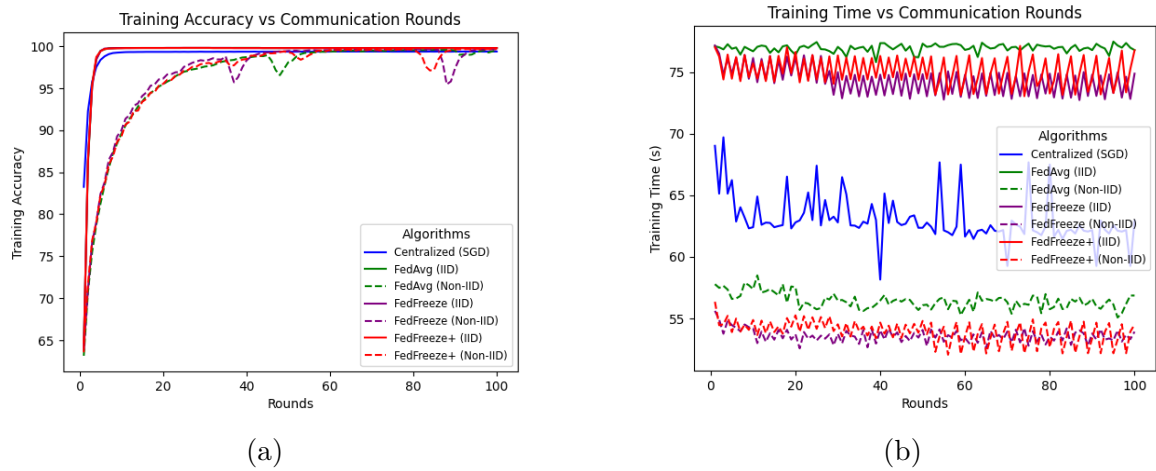


Figure 4.5: (a) Training Accuracy and (b) Training Time Comparison Between Centralized and FL Setup against Local Epoch of $E = 100$ for Centralized and Communication Rounds of $E = 1$, $R = 100$ for FL.

4.3.2 Computation Overhead

Table 4.3 lists the computation cost per round in seconds (S). We observe that FedFreeze and FedFreeze+ execution time is lower than FedAvg in general. Furthermore, we observe that FedFreeze+ takes more time to execute as compared to FedFreeze due the processing time required to freeze the parameters.

4.3.3 Model Accuracy

We present the accuracy of FedFreeze and FedFreeze+ against the centralized and FedAvg as shown in Figure 4.5. Compared to centralized and FedAvg, Figure 4.5a shows that the accuracy of FedFreeze and FedFreeze+ for the IID distribution is closely similar in terms of the convergence gap. We observed that when training the DualL-STMTagger model, FedFreeze+ achieves slightly better accuracy than FedAvg. This is because intermittent parameter freezing breaks down parameter co-adaptations and avoids overfitting, which improves the generalization capability of the model. On the other hand, Figure 4.5b compares the training time of FedFreeze and FedFreeze+ with centralized and FedAvg. In general, FedFreeze and FedFreeze+ exhibit less training time constraints in both the IID and non-IID scenarios. This is due to the use of the freezing technique during training.

Figures 4.6 and 4.7 show the overall performance of FedFreeze and FedFreeze+. Our proposed algorithms show similar performance as FedAvg in the accuracy of the IID scenario as seen in Figures 4.6a and 4.7a, and performed slightly better than FedAvg in the case of the non-IID scenario. On the other hand, the figures show that the FedAvg algorithm exhibits a high and constant data volume because it uploads uncompressed models at the cost of high training time (Figure 4.6b and Figure 4.6c for 10 clients and Figure 4.7b and Figure 4.7c for 30 clients). In contrast, our proposed algorithms fluctuate the data volume in the FL communication phase to reduce the volume of communication data exchanged between the clients and the server. Overall, it is evident that we were able to achieve communication efficiency, but at the expense

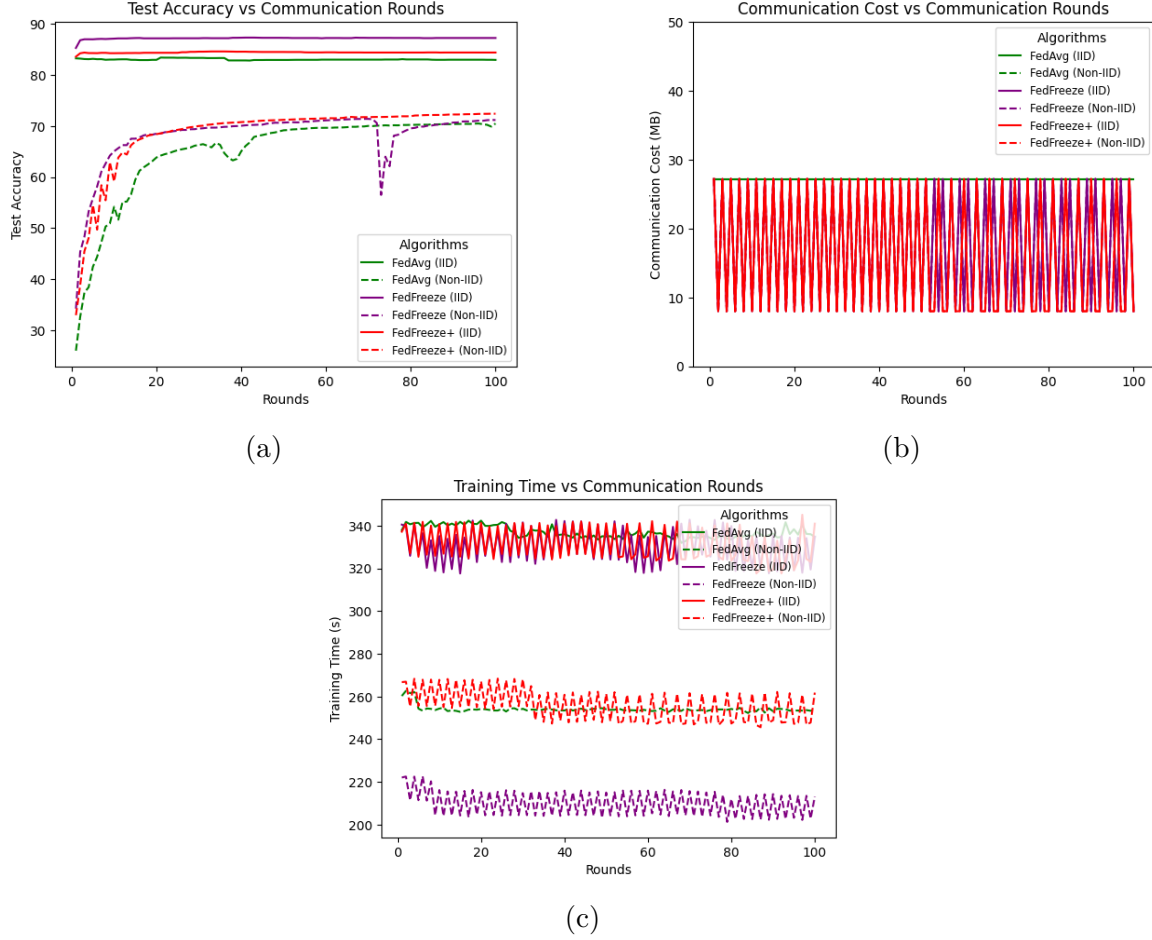


Figure 4.6: Comparison Between FedAvg, FedFreeze & FedFreeze+ on (a) Test Accuracy (b) Communication Cost (c) Training Time for 10 clients with IID and non-IID Distribution.

of a noticeable decrease in the model’s accuracy as the number of participating clients increases.

4.4 Model Deployment

The deployment of the proposed models in the smartphone application yielded impressive performance and outperformed the conventional public NLP model such as the POS tagger in Spacy [85] and NLTK [86] in terms of the ability to handle and

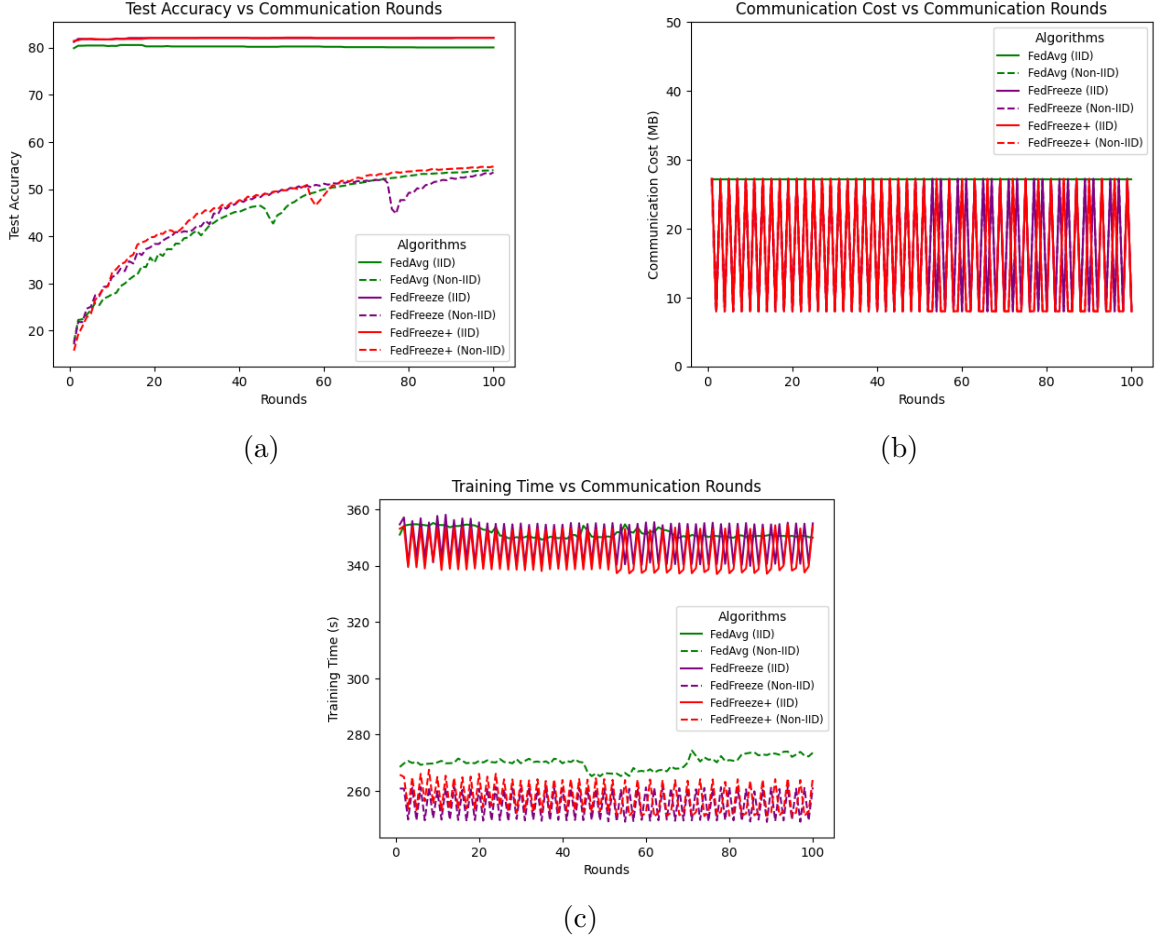


Figure 4.7: Comparison Between FedAvg, FedFreeze & FedFreeze+ on (a) Test Accuracy (b) Communication Cost (c) Training Time for 30 clients with IID and non-IID Distribution.

predict correctly. Figures 4.8 and 4.9 are screenshots of the models deployed on the application. Figure 4.8 presents the complete keyphrase extraction process via the personal assistant application developed for the use case. The figure highlights the stages of speech input during a phone call, the conversion of speech to text, the deployment of our trained model for POS tagging, and based on the POS pattern selected, the correct phrases are extracted. Figure 4.9 shows the model performance in case of context handling and incorrect entry handling.

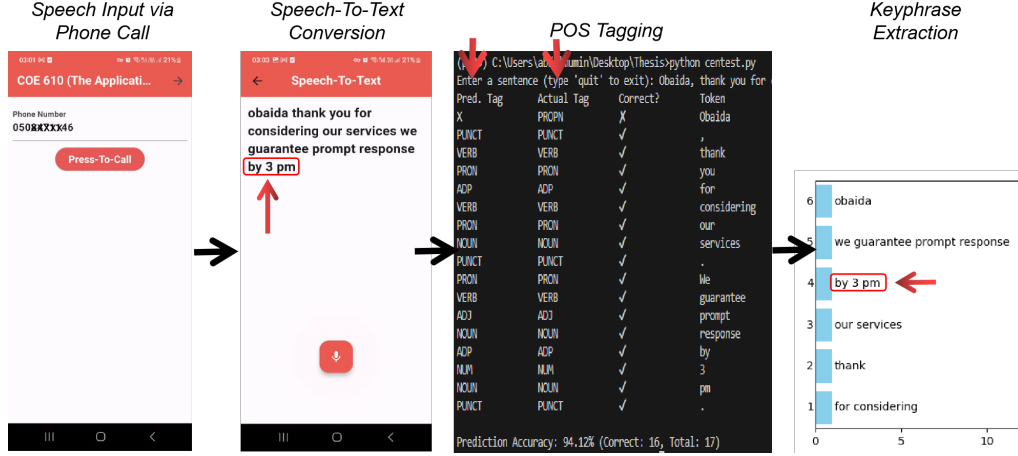


Figure 4.8: *Keyphrase Extraction Demonstration on a Personal Assistant Application.*

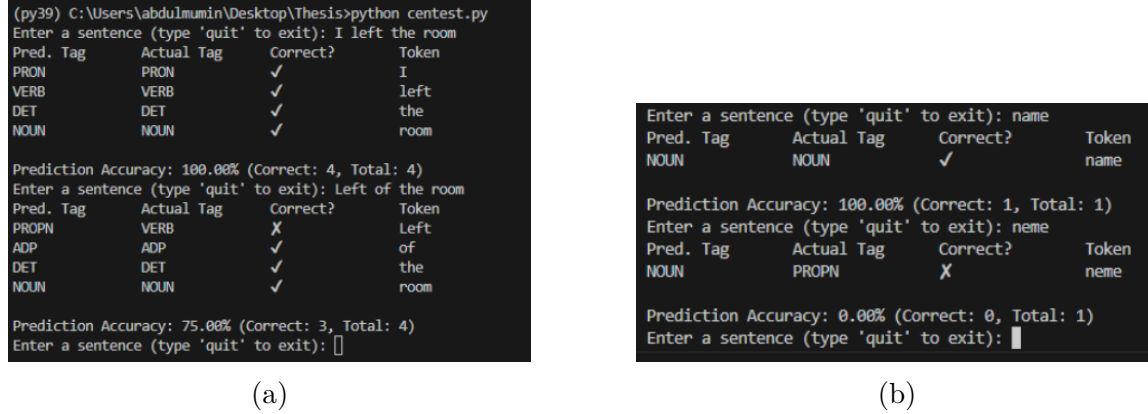


Figure 4.9: (a) Model Prediction and Context Handling. (b) Model Prediction and Wrong Entry Handling.

4.5 Results Summary

In our proposed CEFEL framework, both the FedFreeze and FedFreeze+ algorithms demonstrated a trade-off between communication efficiency, computational overhead, and model accuracy. In non-IID scenarios, our proposed algorithms performed slightly better than FedAvg in test accuracy and computational cost. The algorithms also demonstrated massive improvements in communication efficiency in comparison to FedAvg and yet maintained a similar convergence rate. In general, we observed a

decreasing pattern in model accuracy in all algorithms as the number of participating clients C increases. In addition, from Figures 4.2c and Figure 4.2d we observe that a higher E results in faster convergence without a noticeable benefit in terms of achieving greater test accuracy with a higher E . Figure 4.9 demonstrates the ability to extract keyphrases in the personal assistant application. We observed that the CEFEL framework outperforms traditional public NLP models by accurately handling and predicting based on context, even in the presence of errors or typos.

CHAPTER 5

CONCLUSION

This thesis introduced the CEFEL framework, a federated learning based personal assistant applications on smartphones. The CEFEL framework consists of alternative algorithms, namely; FedFreeze and FedFreeze+ for federated learning environments. The proposed algorithms showed remarkable improvement in reducing communication cost and computational overhead with acceptable accuracy compared to state-of-the-art algorithms, i.e., FedAvg. The proposed algorithms employ parameter freezing in an alternating and incremental fashion. The performance evaluation results demonstrate that we have achieved both communication and computational efficiency in both IID and non-IID scenarios, outperforming standard FL algorithms.

In future work, we intend to investigate the performance of the proposed algorithms in a less constraint environment to obtain thresholds that determine the trade-off points in reducing both communication and computational cost while maintaining an acceptable level of model accuracy. In addition, the deployment of security approaches to the CEFEL framework to deter the growing concern about information recovery, such as Man-in-the-Middle (MITM) attack within the federated space, is part of our

future interest.

REFERENCES

- [1] T. Schopf, S. Klimek, and F. Matthes, “Patternrank: Leveraging pretrained language models and part of speech for unsupervised keyphrase extraction,” *arXiv preprint arXiv:2210.05245*, 2022.
- [2] M. F. Mridha, A. Q. Ohi, M. A. Hamid, and M. Monowar, “A study on the challenges and opportunities of speech recognition for bengali language,” *Artificial Intelligence Review*, vol. 55, no. 4, 2022.
- [3] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, “A survey of federated learning for edge computing: Research problems and solutions,” *High-Confidence Computing*, vol. 1, no. 1, p. 100008, 2021.
- [4] 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.
- [5] M. Ahmadzai and G. Nguyen, “Data partitioning effects in federated learning,” *Proceedings <http://ceur-ws.org> ISSN*, vol. 1613, no. 0073, 2023.
- [6] P. Mane, S. Sonone, N. Gaikwad, and J. Ramteke, “Smart personal assistant using machine learning,” in *2017 International Conference on Energy, Communication,*

Data Analytics and Soft Computing (ICECDS). IEEE, 2017, pp. 368–371.

- [7] I. H. Sarker, A. S. M. Kayes, and P. Watters, “Effectiveness analysis of machine learning classification models for predicting personalized context-aware smart-phone usage,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–28, 2019.
- [8] J. Patil, A. Shewale, E. Bhushan, A. Fernandes, and R. Khartadkar, “A voice based assistant using google dialog flow and machine learning,” *International Journal of Scientific Research in Science and Technology*, vol. 8, no. 3, pp. 6–17, 2021.
- [9] K. Lyons, C. Skeels, T. Starner, C. M. Snoeck, B. A. Wong, and D. Ashbrook, “Augmenting conversations using dual-purpose speech,” in *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, 2004, pp. 237–246.
- [10] P. Cheng and U. Roedig, “Personal voice assistant security and privacy—a survey,” *Proceedings of the IEEE*, vol. 110, no. 4, pp. 476–507, 2022.
- [11] A. R. Javed, M. U. Sarwar, S. ur Rehman, H. U. Khan, Y. D. Al-Otaibi, and W. S. Alnumay, “Pp-spa: privacy preserved smartphone-based personal assistant to improve routine life functioning of cognitive impaired individuals,” *Neural Processing Letters*, pp. 1–18, 2023.
- [12] E. Aiken, S. Bellue, D. Karlan, C. Udry, and J. E. Blumenstock, “Machine learning and phone data can improve targeting of humanitarian aid,” *Nature*, vol. 603, no. 7903, pp. 864–870, 2022.

- [13] J. Patil, A. Shewale, E. Bhushan, A. Fernandes, and R. Khartadkar, “A voice based assistant using google dialogflow and machine learning,” *International Journal of Scientific Research in Science and Technology*, vol. 8, no. 3, pp. 6–17, 2021.
- [14] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein *et al.*, “Federated learning for mobile keyboard prediction,” *arXiv preprint arXiv:1811.03604*, 2018.
- [15] M. M. Qirtas, D. Pesch, E. Zafeiridi, and E. B. White, “Privacy preserving loneliness detection: A federated learning approach,” in *2022 IEEE International Conference on Digital Health (ICDH)*. IEEE, 2022, pp. 157–162.
- [16] J. Zhou, S. Chen, Y. Wu, H. Li, B. Zhang, L. Zhou, Y. Hu, Z. Xiang, Z. Li, N. Chen *et al.*, “Ppml-omics: A privacy-preserving federated machine learning method protects patients’ privacy in omic data,” *Science Advances*, vol. 10, no. 5, p. eadh8601, 2024.
- [17] I. H. Sarker, A. S. M. Kayes, and P. Watters, “Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–28, 2019.
- [18] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1310–1321.

- [19] Y. Zhuansun, D. Li, X. Huang, and C. Sun, “Communication-efficient federated learning with adaptive compression under dynamic bandwidth,” *arXiv preprint arXiv:2405.03248*, 2024.
- [20] K. Sozinov, V. Vlassov, and S. Girdzijauskas, “Human activity recognition using federated learning,” in *2018 IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE, 2018, pp. 1103–1111.
- [21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.
- [22] C. Xu, Y. Qu, Y. Xiang, and L. Gao, “Asynchronous federated learning on heterogeneous devices: A survey,” *Computer Science Review*, vol. 50, p. 100595, 2023.
- [23] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, “Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges,” *IEEE Communications Surveys & Tutorials*, 2023.
- [24] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private recurrent language models,” *arXiv preprint arXiv:1710.06963*, 2017.

- [25] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, “Federated learning with quantization constraints,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8851–8855.
- [26] C. Chen, H. Xu, W. Wang, B. Li, B. Li, L. Chen, and G. Zhang, “Communication-efficient federated learning with adaptive parameter freezing,” in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 1–11.
- [27] V. D. Blondel, A. Decuyper, and G. Krings, “A survey of results on mobile phone datasets analysis,” *EPJ Data Science*, vol. 4, pp. 1–55, 2015.
- [28] N. N. Vo, S. Liu, X. Li, and G. Xu, “Leveraging unstructured call log data for customer churn prediction,” *Knowledge-Based Systems*, vol. 212, p. 106586, 2021.
- [29] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, “Large language models struggle to learn long-tail knowledge,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 15 696–15 707.
- [30] A. Esmailzadeh, M. Heidari, R. Abdolazimi, P. Hajibabaei, and M. Malekzadeh, “Efficient large scale nlp feature engineering with apache spark,” in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022, pp. 0274–0280.
- [31] M. Anand, K. B. Sahay, M. A. Ahmed, D. Sultan, R. R. Chandan, and B. Singh, “Deep learning and natural language processing in computation for offensive lan-

- guage detection in online social networks by feature selection and ensemble classification techniques,” *Theoretical Computer Science*, vol. 943, pp. 203–218, 2023.
- [32] Y. Zhao, X. Jin, and X. Hu, “Recurrent convolutional neural network for speech processing,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5300–5304.
- [33] H. M. Zahera, D. Vollmers, M. A. Sherif, and A.-C. N. Ngomo, “Multpax: Keyphrase extraction using language models and knowledge graphs,” in *International Semantic Web Conference*. Springer, 2022, pp. 303–318.
- [34] M. Song, Y. Feng, and L. Jing, “A survey on recent advances in keyphrase extraction from pre-trained language models,” *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 2153–2164, 2023.
- [35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [36] M. Bansal and T. K. Thivakaran, “Analysis of speech recognition using convolutional neural network,” *Journal of Engineering Sciences*, vol. 11, no. 1, pp. 285–291, 2020.
- [37] D. Sahrawat, D. Mahata, H. Zhang, M. Kulkarni, A. Sharma, R. Gosangi, A. Stent, Y. Kumar, R. R. Shah, and R. Zimmermann, “Keyphrase extraction as sequence labeling using contextualized embeddings,” in *Advances in Information*

- Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42.* Springer, 2020, pp. 328–335.
- [38] M. Rigaki and S. Garcia, “A survey of privacy attacks in machine learning,” *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–34, 2023.
 - [39] M. Khan, F. G. Glavin, and M. Nickles, “Federated learning as a privacy solution—an overview,” *Procedia Computer Science*, vol. 217, pp. 316–325, 2023.
 - [40] J. Liu, J. Huang, Y. Zhou, X. Li, S. Ji, H. Xiong, and D. Dou, “From distributed machine learning to federated learning: A survey,” *Knowledge and Information Systems*, vol. 64, no. 4, pp. 885–917, 2022.
 - [41] 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.
 - [42] X. Yuan and P. Li, “On convergence of fedprox: Local dissimilarity invariant bounds, non-smoothness and beyond,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 10 752–10 765, 2022.
 - [43] T. An, L. Ma, W. Wang, Y. Yang, J. Wang, and Y. Chen, “Consideration of fedprox in privacy protection,” *Electronics*, vol. 12, no. 20, p. 4364, 2023.
 - [44] L. Wang, W. Wang, and B. Li, “Cmfl: Mitigating communication overhead for federated learning,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 954–964.

- [45] A. A. Al-Saedi, V. Boeva, and E. Casalicchio, “Fedco: Communication-efficient federated learning via clustering optimization,” *Future Internet*, vol. 14, no. 12, p. 377, 2022.
- [46] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, “Towards personalized federated learning,” *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [47] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, “A survey on federated learning,” *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.
- [48] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov *et al.*, “Towards federated learning at scale: System design,” *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.
- [49] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, “A survey on security and privacy of federated learning,” *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.
- [50] P. M. Mammen, “Federated learning: Opportunities and challenges,” *arXiv preprint arXiv:2101.05428*, 2021.
- [51] X. Wu, Y. Zhang, M. Shi, P. Li, R. Li, and N. N. Xiong, “An adaptive federated learning scheme with differential privacy preserving,” *Future Generation Computer Systems*, vol. 127, pp. 362–372, 2022.
- [52] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, “Local learning matters: Rethinking data heterogeneity in federated learning,” in *Proceedings of*

- the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8397–8406.
- [53] M. Kim, O. Günlü, and R. F. Schaefer, “Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 2650–2654.
- [54] Z. Zhang, Y. Yang, Y. Dai, Q. Wang, Y. Yu, L. Qu, and Z. Xu, “Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models,” in *Annual Meeting of the Association of Computational Linguistics 2023*. Association for Computational Linguistics (ACL), 2023, pp. 9963–9977.
- [55] J. So, C. He, C. S. Yang, S. Li, Q. Yu, R. E. Ali, and S. Avestimehr, “Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning,” in *Proceedings of Machine Learning and Systems*, vol. 4, 2022, pp. 694–720.
- [56] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, and K. Seth, “Practical secure aggregation for federated learning on user-held data,” *arXiv preprint arXiv:1611.04482*, 2016.
- [57] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, “Personalized federated learning with differential privacy,” *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9530–9539, 2020.

- [58] J. Li, Y. Meng, L. Ma, S. Du, H. Zhu, Q. Pei, and X. Shen, “A federated learning based privacy-preserving smart healthcare system,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 2244–2254, 2021.
- [59] Q. Wang, H. Yin, T. Chen, J. Yu, A. Zhou, and X. Zhang, “Fast-adapting and privacy-preserving federated recommender system,” *The VLDB Journal*, vol. 31, no. 5, pp. 877–896, 2022.
- [60] L. Yuan, Z. Wang, L. Sun, S. Y. Philip, and C. G. Brinton, “Decentralized federated learning: A survey and perspective,” *IEEE Internet of Things Journal*, 2024.
- [61] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, “A survey on federated learning: Challenges and applications,” *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 2, pp. 513–535, 2023.
- [62] V. Smith, C. K. Chiang, M. Sanjabi, and A. S. Talwalkar, “Federated multi-task learning,” in *Advances in neural information processing systems*, vol. 30, 2017.
- [63] S. Gupta, Y. Huang, Z. Zhong, T. Gao, K. Li, and D. Chen, “Recovering private text in federated learning of language models,” *Advances in neural information processing systems*, vol. 35, pp. 8130–8143, 2022.
- [64] K. Pillutla, K. Malik, A.-R. Mohamed, M. Rabbat, M. Sanjabi, and L. Xiao, “Federated learning with partial model personalization,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 17 716–17 758.

- [65] J. Konečný, 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.
- [66] X. Li, Z. Qu, B. Tang, and Z. Lu, “Fedlga: Toward system-heterogeneity of federated learning via local gradient approximation,” *IEEE Transactions on Cybernetics*, 2023.
- [67] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “Freezeout: Accelerate training by progressively freezing layers,” *arXiv preprint arXiv:1706.04983*, 2017.
- [68] J. Wu, T. Dai, P. Guan, S. Liu, F. Gou, A. Taherkordi, Y. Li, and T. Li, “Fedapt: Joint adaptive parameter freezing and resource allocation for communication-efficient federated vehicular networks,” *IEEE Internet of Things Journal*, 2024.
- [69] K. Pfeiffer, M. Rapp, R. Khalili, and J. Henkel, “Cocofl: Communication-and computation-aware federated learning via partial nn freezing and quantization,” *arXiv preprint arXiv:2203.05468*, 2022.
- [70] Y. Wang, L. Lin, and J. Chen, “Communication-efficient adaptive federated learning,” in *International Conference on Machine Learning*. PMLR, June 2022, pp. 22 802–22 838.
- [71] X. Xiao, T. B. Mudiyansele, C. Ji, J. Hu, and Y. Pan, “Fast deep learning training through intelligently freezing layers,” in *2019 international conference on internet of things (iThings) and IEEE green computing and communications*

- (*GreenCom*) and *IEEE cyber, physical and social computing (CPSCoM)* and *IEEE smart data (SmartData)*. IEEE, 2019, pp. 1225–1232.
- [72] C. Chen, H. Xu, W. Wang, B. Li, B. Li, L. Chen, and G. Zhang, “Synchronize only the immature parameters: Communication-efficient federated learning by freezing parameters adaptively,” *IEEE Transactions on Parallel and Distributed Systems*, 2023.
- [73] E. Malan, V. Peluso, A. Calimera, E. Macii, and P. Montuschi, “Automatic layer freezing for communication efficiency in cross-device federated learning,” *IEEE Internet of Things Journal*, 2023.
- [74] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, “Federated learning with matched averaging,” *arXiv preprint arXiv:2002.06440*, 2020.
- [75] J. Pei, K. Zhong, M. A. Jan, and J. Li, “Personalized federated learning framework for network traffic anomaly detection,” *Computer Networks*, vol. 209, p. 108906, 2022.
- [76] B. Y. Lin, C. He, Z. Zeng, H. Wang, Y. Huang, C. Dupuy, R. Gupta, M. Soltanolkotabi, X. Ren, and S. Avestimehr, “Fednlp: Benchmarking federated learning methods for natural language processing tasks,” *arXiv preprint arXiv:2104.08815*, 2021.
- [77] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, and R. Lowe, “Training language models to follow instructions with human feed-

- back,” in *Advances in neural information processing systems*, vol. 35, 2022, pp. 27 730–27 744.
- [78] O. S. Prabhu, P. K. Gupta, P. Shashank, K. Chandrasekaran, and D. Usha, “Towards a federated learning approach for nlp applications,” in *Applications of Artificial Intelligence and Machine Learning: Select Proceedings of ICAAAIML 2020*. Springer Singapore, 2021, pp. 157–167.
- [79] D. Cai, Y. Wu, S. Wang, F. X. Lin, and M. Xu, “Efficient federated learning for modern nlp,” in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, October 2023, pp. 1–16.
- [80] J. Deng, C. Wang, X. Meng, Y. Wang, J. Li, S. Lin, and C. Ding, “A secure and efficient federated learning framework for nlp,” *arXiv preprint arXiv:2201.11934*, 2022.
- [81] F. Jiang, L. Dong, S. Tu, Y. Peng, K. Wang, K. Yang, C. Pan, and D. Niyato, “Personalized wireless federated learning for large language models,” *arXiv preprint arXiv:2404.13238*, 2024.
- [82] O. Galal, A. H. Abdel-Gawad, and M. Farouk, “Federated freeze bert for text classification,” *Journal of Big Data*, vol. 11, no. 1, p. 28, 2024.
- [83] A. Zeldes, “The GUM corpus: Creating multilayer resources in the classroom,” *Language Resources and Evaluation*, vol. 51, no. 3, pp. 581–612, 2017.

- [84] V. Makarenkov, L. Rokach, and B. Shapira, “Choosing the right word: Using bidirectional lstm tagger for writing support systems,” *Engineering Applications of Artificial Intelligence*, vol. 84, pp. 1–10, 2019.
- [85] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, “spaCy: Industrial-strength Natural Language Processing in Python,” 2020.
- [86] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O’Reilly Media, Inc., Jun. 2009. [Online]. Available: <https://www.nltk.org/book/>

VITAE

- Name: Abdulmumin Sa'ad
- Nationality: Nigerian
- Date of Birth: 25/01/1994
- Email: *abdulmuminsaad94@gmail.com*
- Permenant Address: No. 11 Taura Close, Kubwa NNPC, FCT Abuja, 900108, Nigeria
- Education:
 - **King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia** (Master of Science in Computer Engineering) Aug 2022-Jun 2024
 - **Bayero University Kano, Nigeria** (Bachelor of Engineering in Computer Engineering) Dec 2015-Jan 2020
 - **Kaduna Polytechnic, Nigeria** (National Diploma in Computer Engineering) Jan 2012-Oct 2015

- Conferences & Certifications:
 - Participant at 1st KFUPM Interdisciplinary Graduate Students Conference, (KFUPM, 2024)
 - Cisco Cyber Operations (Cisco Networking Academy, University of Jos, 2020)
 - Certified Network Security Specialist (International Cyber Security Institute, 2020)
 - Python Certified Associate Programmer (Python Institute, 2019)
 - Cisco Certified Network Associate, (Cisco Networking Academy, University of Jos, 2019)
- Awards & Recognition:
 - Emerged 3rd place in KFUPM-AWS Deep Racer League, KFUPM, Saudi Arabia (2024)
 - Fully Funded International MSc Scholarship Recipient, KFUPM, Saudi Arabia (2022)
 - Active Instructor Award, Cisco Networking Academy (2022)
 - Member, Nigerian Computer Society, (2021)
 - Member, Nigerian Society of Engineers, (2021)
 - Best Student Per Semester, BUK Nigeria (2016, 2017, 2018)
 - Engineering Scholar of the Year, Nigeria Higher Education Foundation (2018)