KORRAPATI, BHUVANA, M.S. Enhancing Federated Learning Efficiency through Dynamic Model Adaptation and Optimization. (2024)
Directed by Dr. Jing Deng. 88 pp.

Federated learning (FL) in cloud computing has emerged as a groundbreaking paradigm, revolutionizing data processing and machine learning through decentralized and scalable systems. However, the integration of these technologies faces challenges in communication efficiency and data privacy preservation, which are crucial for their widespread adoption and effectiveness. This research presents a dynamic federated learning approach that incorporates model compression, PCA-based dimensionality reduction, and fine-tuning to address these challenges.

The proposed method dynamically determines the optimal number of PCA components based on client data variability, effectively reducing data dimensionality. By applying model compression techniques, including pruning and quantization, the approach enhances communication efficiency without compromising the performance. Furthermore, the integration of fine-tuning as a knowledge distillation step allows the compressed models to adapt to client-specific data patterns, thereby tackling issues of skewness and overfitting.

In addressing the challenges of bandwidth and latency in FL, the evaluation metrics encompass Average Communication Cost, Average Bandwidth Utilization, and Average Latency, demonstrating the approach's effectiveness in optimizing these key performance indicators. Moreover, the framework incorporates dynamic model adaptation on both client-side and server-side, enabling personalized adjustments based on local data characteristics and client resources, while optimizing the global model's performance.

Using both the MNIST and CIFAR-10 datasets for validation, this approach

demonstrates maintained accuracy with data reduction across various levels of data skewness and complexity. The proposed federated learning framework follows a comprehensive flow chart that encompasses server initialization, model distribution, client-side processing (including data dimensionality reduction, model compression, local training, fine-tuning, and dynamic adaptation), server-side aggregation, global model update, model evaluation, and iterative refinement. This research contributes to the advancing field of cloud-based FL by presenting an efficient, privacy-preserving, and scalable approach for distributed machine learning, setting a new standard for optimizing communication efficiency in decentralized data environments and paving the way for the next generation of federated learning systems that prioritize efficiency, privacy, and scalability.

ENHANCING FEDERATED LEARNING EFFICIENCY THROUGH DYNAMIC MODEL ADAPTATION AND OPTIMIZATION

by

Bhuvana Korrapati

A Thesis Submitted to
the Faculty of The Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Greensboro
2024

Approved by

_____
Committee Chair

To my parents for their continual support of my academic career.

APPROVAL PAGE

This thesis written by Bhuvana Korrapati has been approved by the following committee of the Faculty of The Graduate School at The University of North Carolina at Greensboro.

Committee Chair _____
Jing Deng

Committee Members _____
Minjeong Kim

_____
Qianqian Tong

_____
Date of Acceptance by Committee

_____
Date of Final Oral Examination

ACKNOWLEDGMENTS

I would like to thank my advisor, Jing Deng, for helping me to prepare this Thesis. His experience and advise were invaluable.

PREFACE

In this research, we explore the dynamic integration of Federated Machine Learning (FL) and cloud computing, focusing particularly on optimizing communication efficiency through adaptive model modifications. Our study introduces innovative methodologies that include dynamic data reduction and model compression techniques, tailored to the variability in client data. These strategies are vital for enhancing the scalability and performance of FL systems across heterogeneous cloud environments. The adaptive nature of our approach not only enriches the academic landscape but also lays a robust foundation for developing secure, efficient, and practical FL implementations that are responsive to real-world complexities.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

In the introduction, we start by giving a brief overview of our project, outlining its scope and objectives. We then explain our decision to use the Federated Learning approach, highlighting its significance to our research. Lastly, we identify the main communication challenges with data transfer and encountered in Federated Learning, setting the stage for our exploration of these issues.

I.1. Overview

The integration of Federated Machine Learning (FL) with cloud computing marks a pivotal development in the realms of data science and artificial intelligence. This overview endeavors to illuminate the core principles underlying both FL and cloud computing, delineating their distinct contributions and synergy within contemporary computational paradigms. Central to this discourse is the endeavor to optimize communication efficiency through model reduction, a critical determinant for the practicality and effectiveness of FL within cloud infrastructures.

Federated Machine Learning is a novel approach that deviates from traditional centralized machine learning methods. In FL, the learning process is distributed across a multitude of devices or servers, each holding its own local data set. This decentralized structure allows for the simultaneous training of machine learning models on numerous nodes, which then communicate their findings or model updates to a central server. The primary advantage of this approach is its ability to leverage diverse data sources while maintaining the confidentiality and integrity of each dataset and it is used in the sectors such as healthcare and finance.

Cloud platforms offer scalable, reliable, and accessible computational resources, enabling the efficient handling and processing of data and model updates from various nodes in the FL network. The cloud acts as a central point where aggregated learning takes place, synthesizing insights from distributed data sources without the need for data centralization.

The synergy between FL and cloud computing is pivotal in overcoming some of the traditional limitations of machine learning models, particularly those related to data privacy, scalability, and resource optimization. While FL ensures that data remains at its source, thereby preserving privacy and reducing data movement, cloud computing provides the necessary infrastructure for effective model aggregation and global update. This relationship not only enhances the efficiency of machine learning processes but also opens up new avenues for leveraging big data in a privacy-preserving and resource-efficient manner.

I.2. Federated Vs Traditional approach

Federated Machine Learning (FL) is chosen over traditional machine learning approaches primarily due to its distributed nature as opposed to the centralized model of conventional learning. In traditional machine learning, data from various sources is aggregated in a central location for model training, posing challenges in data privacy and scalability. In contrast, FL allows for the model to be trained across a network of decentralized devices, each working with its own local data. This key distinction enables FL to leverage the diverse data sets available across various nodes while maintaining data at its source, thus addressing the centralization concerns inherent in traditional machine learning methods.

The choice of Federated Learning (FL) over traditional machine learning approaches is driven by several compelling reasons which includes Data Privacy and

Security, Bandwidth Efficiency, Compliance with Data Regulations, User Preference, Data volume, Real-world Data Representation and Resource Optimization. To deal with these drawbacks we have chosen to use federtated machine learning.

(a) Data Privacy and Security: FL enables machine learning models to be trained across multiple devices or nodes while keeping the data localized. This means sensitive or proprietary data doesn't need to be shared or transferred to a central server, significantly enhancing data privacy and security. This is particularly important in industries where data confidentiality is crucial, like healthcare or finance.

(b) Bandwidth Efficiency: Traditional machine learning often requires transferring large datasets to a central location for processing, which can be bandwidth-intensive and impractical, especially with large-scale data. FL, by processing data locally and only exchanging model updates, significantly reduces bandwidth requirements.

(c) Regulations: GDPR (Europe), CCPA (California), PIPEDA (Canada), LGPD (Brazil), PDPL (Argentina), KVKK (Turkey), POPI (South Africa), FSS (Russia), CDPR (China), PDPB (India), PIPA (Korea), APPI (Japan), PDP (Indonesia), PDPA (Singapore), APP (Australia), and other regulations protect data from being moved. In fact, those regulations sometimes even prevent single organizations from combining their own users' data for AI training because those users live in different parts of the world, and their data is governed by different data protection regulations.

(d) User preference: In addition to regulation, there are use cases where users just expect that no data leaves their device, ever. If you type your passwords and credit card info into the digital keyboard of your phone, you don't expect those passwords to end up on the server of the company that developed that keyboard, do you? In fact, that use case was the reason federated learning was invented in the first place.

(e)Data-Volume: Some sensors, like cameras, produce such a high data volume that

it is neither feasible nor economic to collect all the data. Think about a national rail service with hundreds of train stations across the country. If each of these train stations is outfitted with a number of security cameras, the volume of data they produce requires incredibly powerful and exceedingly expensive infrastructure to process and store. And most of the data isn't even useful.

(f) Real-world Data Representation: In traditional centralized machine learning, the model is often trained on a homogeneous dataset, which might not represent real-world scenarios accurately. FL, by training on diverse, decentralized data, can result in models that are more generalizable and effective in real-life conditions.

(g) Resource Optimization: FL allows for leveraging the computational resources of client devices, distributing the workload, and thereby reducing the computational burden on a single central server. This is especially beneficial when dealing with large networks of devices.

I.3. Centralized Learning Vs Federated Learning

Federated learning inverts the traditional method by bringing the model training to where the data resides, rather than centralizing data for training. This approach allows:

Traditional learning centralizes data for analysis.whereas, Federated learning decentralizes the process, taking analysis to the data sources.

This shift opens new possibilities for applying machine learning in fields previously restricted by data privacy or logistical issues. It paves the way for collaborative advancements in healthcare, financial security, and privacy-focused technologies, among others, by tapping into the wealth of data available across various institutions. As federated learning expands, it unveils fresh opportunities for innovation by leveraging data that was once unreachable.

(a) Centralized Learning Approach: The Centralized Learning Approach, a cornerstone of traditional machine learning, hinges on consolidating data from diverse sources such as mobile users or IoT sensors into a central repository for processing, usually hosted on servers or cloud systems. This method capitalizes on the richness of the combined dataset to train models that extract broad insights and patterns, reflecting the aggregate knowledge.

Upon gathering this data, the central hub leverages its substantial computational power to refine a machine learning model, which assimilates the collective intelligence embedded in the data. Post-training, the model is deployed for predictive tasks, either directly on the server for immediate API access or redistributed to clients for localized use.

However, this model's centralization of data aggregation and storage introduces critical privacy and security dilemmas. The concentration of diverse client data in a single locale not only heightens the risk of exposing sensitive information but also turns the central data repository into a prime target for cyber threats and data breaches, amplifying concerns over data misuse and security vulnerabilities.

Moreover, the reliance on a centralized infrastructure can lead to scalability issues as the system must handle increasing volumes of data and computation. This can result in bottlenecks, where the capacity to process and analyze data efficiently becomes constrained by the central server's limitations. Additionally, the need for continuous data transmission to the central server can exacerbate network congestion, leading to delays and reduced system responsiveness, further challenging the scalability and efficiency of centralized learning systems.

(b) Federated Learning Approach: Federated learning begins with initializing a model on a central server, similar to traditional machine learning setups. This

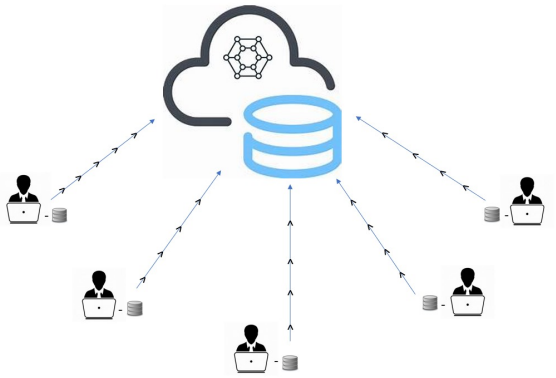

Figure I.1. Centralized Learning Approach

model's parameters, either freshly initialized or based on a previous model, are distributed to selected participant nodes, like devices or institutions, ensuring a uniform starting point for all.

These nodes then proceed to train the model locally with their unique datasets, but only for a short duration, which might range from a single epoch to a few mini-batch steps. Following this local training phase, each node ends up with a personalized version of the model, reflecting the peculiarities of its data.

The nodes then transmit their model updates back to the central server. These updates can be the complete model parameters or just the gradients from the local training. With updates from, say, 10 nodes, the server faces the challenge of synthesizing these diverse learnings into a single, coherent model.

This is where the aggregation step comes into play, with Federated Averaging (FedAvg) being a common method. It involves taking a weighted average of the updates, where the weights correspond to the volume of data used by each node, ensuring equitable representation of each dataset in the global model.

This cycle, comprising the distribution of the global model, local training by

nodes, and aggregation of updates, constitutes a single round of federated learning. Repeating these rounds enhances the model's performance across the collective data of all nodes, culminating in a robust, widely applicable model.

Beyond model training, federated learning also encompasses federated evaluation, allowing for the performance assessment of the model across diverse datasets. Additionally, federated analytics can offer insights into broader trends without compromising individual data privacy, thanks to techniques like secure aggregation.

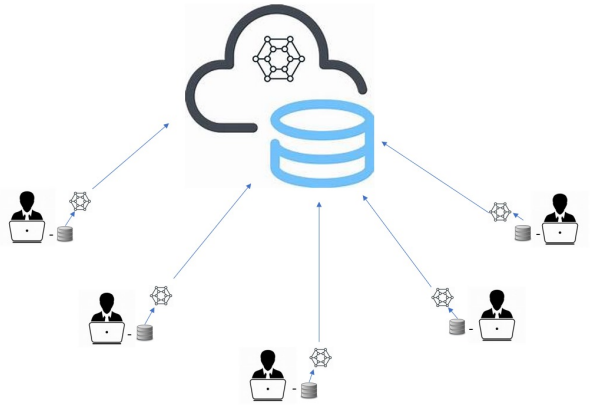

Figure I.2. Federated Learning Approach

CHAPTER II

RELATED WORK

In this chapter, we showcase the broad applications of Federated Learning across multiple disciplines, detailing its intersection with fields such as data privacy, model robustness, and computational efficiency. We review seminal contributions addressing these Federated Learning challenges and their integration with cloud computing. Further, we demonstrate how our research extends beyond existing efforts, offering innovative advancements to the field.

II.1. Federated Learning Applications

"Federated Learning Applications" delves into the transformative impact of Federated Learning across a spectrum of sectors, from Healthcare's patient data privacy to Mobile Computing's device-centric models. In Self-Driving Cars, it enhances real-time decision-making while safeguarding sensitive data. The Internet of Things benefits from decentralized data processing, and Information Technology sees improved security and efficiency in data management. This exploration showcases the versatility and potential of Federated Learning in addressing industry-specific challenges.

II.1.1 Healthcare

Federated Learning in healthcare as in the research paper [1] represents a groundbreaking shift towards more secure and efficient data utilization, addressing the critical challenge of data fragmentation and privacy. By enabling disparate healthcare institutions to collaboratively train machine learning models without sharing sensitive patient data, Federated Learning ensures robust, generalizable insights across diverse

populations. This approach is particularly beneficial in creating comprehensive models from Electronic Health Records (EHR), enhancing diagnostic accuracy and personalized care while upholding stringent privacy standards. Moreover, integrating Federated Learning with edge computing opens new avenues for real-time, on-site data processing, further improving response times and patient outcomes in critical care scenarios, such as COVID-19 diagnosis [2]. This synergistic use of Federated Learning and edge computing in healthcare not only surpasses traditional cloud-based solutions in privacy and latency but also marks a significant leap in harnessing the full potential of distributed data for advancing medical research and care delivery.

II.1.2 Self-Driving Cars

The Federated Learning approach is revolutionizing the automation sector, particularly in advancing self-driving car technologies [3]. By enabling on-device machine learning, this approach ensures real-time data processing directly at the edge, significantly enhancing the responsiveness and safety of autonomous vehicles. For instance, in steering angle prediction, Federated Learning has demonstrated its capability to refine the accuracy of local models to match centralized training, without the associated privacy and efficiency drawbacks. Moreover, the introduction of Deep Federated Learning (DFL) and specialized networks like Federated Autonomous Driving network (FADNet) further underscores the potential of decentralized learning in handling complex tasks such as autonomous driving policy formulation[4]. These advancements not only accelerate the development cycle by reducing communication overheads but also address privacy concerns by eliminating the need for data centralization. Thus, Federated Learning is setting new benchmarks in the automation industry, offering a scalable, privacy-conscious framework for the continuous improvement of autonomous driving systems.

II.1.3 Mobile edge Computing

Federated Learning (FL) in the realm of Mobile Edge Computing (MEC) represents a promising avenue for deploying deep learning applications directly at the network's edge, enhancing privacy, reducing latency, and alleviating bandwidth constraints[5]. The first research paper delves into the integration of FL with MEC, spotlighting the resource-intensive challenges such as computation, bandwidth, energy, and data faced by mobile clients. It categorizes resource optimization strategies into black-box approaches, including training adjustments and client selection, and white-box approaches, like model compression and asynchronous updates. The paper proposes a novel neural-structure-aware resource management technique, leveraging module-based FL to tailor global model subnetworks to the resource capacities of individual mobile clients, showcasing significant improvements in resource efficiency.

The research paper [6] focuses on the burgeoning field of Augmented Reality (AR), highlighting the critical need for high data rates and low latency in AR applications. It addresses the computational and latency challenges inherent in AR object detection and classification by proposing a synergistic framework that combines MEC with FL. This decentralized approach significantly reduces the need for data transmission and processing bandwidth, while also decreasing the number of training iterations required when compared to traditional centralized learning methods. The framework's effectiveness is validated using the CIFAR-10 dataset, illustrating the potential of FL in MEC to enhance AR applications' performance and user experience.

II.1.4 Internet of Things

The advent of billions of IoT devices, propelled by the rapid advancements in 5G/6G technologies, heralds a new era of connectivity and data generation,

presenting both opportunities and challenges in data privacy and system efficiency. The paper [7] discusses the transformative potential of Federated Learning (FL) as a solution to the privacy and efficiency challenges posed by the traditional centralized data processing models in IoT ecosystems. FL enables collaborative, data-driven model training across multiple IoT devices without centralizing data, thus significantly reducing communication and storage demands while enhancing user privacy. However, the implementation of FL in IoT networks faces hurdles, including seven identified critical challenges that demand innovative solutions to fully harness FL's potential in diverse IoT applications.

The paper [8] further explores the integration of FL within IoT, emphasizing its role in addressing the privacy and distribution challenges inherent in IoT's vast, decentralized networks. By facilitating on-device machine learning and limiting data exchange to model updates, FL offers a promising avenue for privacy-preserving, efficient network and application management in IoT. The paper provides a comprehensive overview of recent FL advancements tailored for IoT, including various metrics for evaluating FL's performance and a taxonomy for its application over IoT networks. It also outlines several open research challenges and potential solutions, underscoring the dynamic and evolving nature of FL in the IoT domain. Together, these papers highlight the critical role of FL in advancing IoT technologies while navigating the complex landscape of privacy, efficiency, and scalability challenges.

II.1.5 Information Technology

This paper [9] offers a deep dive into Federated Learning (FL) within Information Technology, highlighting its role in developing secure, privacy-first data solutions across industries. FL's collaborative model allows for algorithm training across decentralized nodes without direct data sharing, enhancing privacy and

security. The study covers FL's foundational aspects, key enabling technologies, and practical applications, aiming to arm data scientists with the insights needed for deploying FL in privacy-sensitive environments. It also outlines the main challenges and real-life use cases, showcasing FL's potential to revolutionize data privacy and security practices.

II.2. Federated Learning privacy

Federated Learning stands at the forefront of privacy-preserving machine learning, addressing critical concerns through innovative approaches like Differential Privacy, Secure Multi-Party Computation, and Federated Transfer Learning. While FL inherently enhances data privacy by training models across decentralized devices without sharing raw data, it remains susceptible to privacy attacks. DP introduces noise to data or updates to obscure individual contributions, SMPC enables secure, collaborative computation, and FTL facilitates knowledge transfer across domains without direct data exchange, each adding layers of privacy protection against potential vulnerabilities.

II.2.1 Privacy Attacks

The literature presents various instances where Federated Learning (FL) is vulnerable to privacy breaches. For instance, the mGAN-AI framework [10] employs a Generative Adversarial Network (GAN) to undermine client privacy by synthesizing class representatives from local updates, revealing sensitive client data characteristics. Similarly, Nasr et al. [11] showcased how membership inference attacks could ascertain a data point's participation in model training by analyzing its influence on the model's features, thus compromising individual data privacy. Melis et al. [12] revealed the feasibility of property inference attacks, where attackers discern specific attributes in training data by comparing model updates derived from datasets with

and without the desired attributes. Furthermore, the Deep Leakage from Gradients attack [13] demonstrates the potential for attackers to reconstruct private training samples and labels by iteratively adjusting dummy data to align with the actual model updates, posing a significant threat to data confidentiality in FL environments.

II.2.2 Differential Privacy(DP)

Differential Privacy (DP) serves as a critical line of defense in safeguarding Federated Learning (FL) systems. Geyer et al. [14] introduced a mechanism that enhances the privacy of global model updates by altering the server's aggregation method. This involves clipping individual updates based on Euclidean distance to cap the extractable information from each, followed by the addition of Gaussian noise to the aggregate of these clipped updates, thereby diluting the gleanable insights from the collective updates. The aggregated, noised update is then normalized by the total number of contributors to produce the final adjusted global model update. While this approach offers protection against inference attacks targeting global updates, it predicates on the server's trustworthiness due to its access to local updates. The introduction of noise, albeit enhancing privacy, can detract from model accuracy. However, as evidenced by McMahan et al. [15] , an increase in participant count can mitigate this impact, making the strategy particularly viable for cross-device FL scenarios with numerous clients. Additionally, this method allows for the application of poisoning defense mechanisms, given the server's visibility into client updates, unlike some other privacy-preserving techniques.

II.2.3 Secure Multi-Party Computation (SMPC)

In the context of secure multi-party computation (SMPC) within federated learning (FL)[16] , this document highlights the integration of simulation-based privacy-preserving mechanisms, as discussed in the work by Kairouz et al. It delves

into how FL, when framed within a simulation-based approach, leverages the principles of SMPC to safeguard the privacy and security of the computed data. By adopting techniques inherent to SMPC, such as homomorphic encryption (HE) and differential privacy (DP), FL is able to ensure that the m-ary functionality, which it aims to compute, remains confidential and secure against potential adversarial threats. This approach underscores the efficacy of simulation-based frameworks in enhancing the privacy aspects of FL, thereby positioning simulation-based federated learning (SFL) as a specialized subset that inherently embodies the core principles of secure multi-party computation. This perspective not only enriches the understanding of FL's privacy-preserving capabilities but also reinforces its alignment with the rigorous standards of SMPC, highlighting the synergy between FL and SMPC in advancing secure and privacy-conscious distributed computing.

II.2.4 Federated Transfer Learning (FTL)

The research paper[17] introduces Federated Transfer Learning (FTL), a novel approach designed to navigate the challenges of data fragmentation across different organizations while adhering to privacy and legal constraints. FTL facilitates the sharing of knowledge and transfer of insights across domains without compromising user privacy, enabling the construction of robust statistical models by leveraging labeled data from a source domain. This innovative framework integrates seamlessly with existing models, maintaining accuracy levels comparable to traditional transfer learning methods, and is versatile enough to be applied to a variety of secure multi-party machine learning tasks.

The paper[18] delves into the application of federated transfer learning within the context of smart manufacturing, aiming to address the dual challenges of data scarcity and privacy preservation in machine learning. By proposing a federated

transfer learning framework for cross-domain prediction, the study showcases how knowledge from existing applications can be shared and adapted to new manufacturing processes and products through a central server and smart devices network. This framework enhances model accuracy through federated learning within groups and ensures data privacy by avoiding raw data exposure. The effectiveness of this method is demonstrated through superior performance in learning efficiency and accuracy on public datasets like COCO and PETS2009, compared to other leading machine learning approaches.

II.3. Federated Learning Communication Efficiency

In the section on with reference to the different research publications on communication efficiency involving federated learning approach, we explore innovative strategies aimed at enhancing the efficiency of communication in federated learning environments. This section delves into two pivotal subparts.

II.3.1 Federated Dropout

Federated Dropout, introduced in research [19], innovatively minimizes server-to-client communication costs by focusing on training smaller, sub-model segments of the overarching model. This strategy leverages compression techniques, such as quantization, to enhance efficiency further. Initially, the server delineates a subset of the global model, applies compression, and dispatches this condensed version to the client. Upon receipt, the client expands this update, integrates it with local data for training, and then recalculates its local update. This newly derived update is then compressed by the client for transmission back to the server. Upon collecting these compressed updates from various clients, the server decompresses them and conducts an aggregation process to synthesize the updated global model. This method not only conserves bandwidth but also enables scalable and efficient

model training across a broad network of distributed clients.

II.3.2 Structured and Sketched updates

Structured and Sketched updates, as delineated in research [20], introduce significant optimizations to enhance the efficiency of client-to-server communications in Federated Learning environments. The concept of Structured updates mandates that local updates adhere to a predefined structure, such as being a low-rank or sparse matrix, which inherently reduces the volume of parameters required for transmission, thereby optimizing bandwidth usage. On the other hand, Sketched updates involve the compression of local updates post-training through methods like subsampling or quantization. This technique not only streamlines the data transmission process but also contributes to reducing the computational load on the network, making the overall Federated Learning process more efficient and scalable. These optimizations collectively address critical challenges in FL, particularly in scenarios involving constrained network resources or extensive client networks.

II.4. Communication Challenges in Federated Learning

This section overviews significant contributions to communication reduction in federated learning, setting the stage for discussing the distinctiveness of the proposed model based on earlier provided code details.

II.4.1 FedSCR: Structure-Based Communication Reduction for Federated Learning

FedSCR [21]employs a structure-based communication reduction algorithm that identifies negligible updates to reduce the number of parameters transmitted, optimizing network use without compromising accuracy. It aggregates updates over channels and filters, which helps maintain the complete model structure without retraining or fine-tuning. This approach is particularly beneficial in environments

with unbalanced data distribution.

II.4.2 COFIG and FRECON

The introduction of COFIG and FRECON targets the dual challenges of high communication costs and client variance in federated settings [22].COFIG and FRECON address communication overhead and client variance by reducing the number of communication rounds required for convergence in federated learning. These methods provide convergence proofs in nonconvex settings, showcasing substantial efficiency improvements. They do not require communication with all clients in each round, which enhances scalability and reduces network load.

Two-Stream Federated Learning The Two-Stream Federated Learning [**?**]. method incorporates a Maximum Mean Discrepancy (MMD) constraint to efficiently handle Non-IID data distributions, reducing communication rounds by more than 20%. This approach optimizes data handling in complex data environments but involves maintaining dual model architectures.

II.4.3 Structured and Sketched Updates

This technique explores the reduction of uplink communication costs through structured learning updates, demonstrating significant reductions in the amount of data transmitted [23]. FedSCR complements this by not only reducing the volume of data sent but also ensuring that the reductions do not compromise the accuracy of the model, thanks to its adaptive thresholding which adjusts to the model's real-time learning needs.

Deep Compression for OTA-FL Deep Compression techniques applied in Over-the-Air Federated Learning (OTA-FL) dramatically decrease both computational and communication demands by integrating pruning and quantization-aware training [24].

II.4.4 Adaptive Dynamic Pruning for Non-IID FL

Adaptive Dynamic Pruning addresses the unique challenges posed by Non-IID datasets on edge devices, enhancing inference speeds significantly [25]. FedSCR builds on this premise by not only accelerating inference but also reducing communication overhead through its novel aggregation and reduction algorithm, which dynamically adapts to data variability across clients. This dual focus on speed and communication efficiency.

II.4.5 Our Proposed Model

Our proposed model, built on insights from existing methods, introduces a novel adaptive thresholding mechanism that dynamically optimizes communication based on real-time data evaluations. Unlike FedSCR, which statically aggregates parameter updates, our model dynamically adjusts to variations in data distribution, enhancing robustness and efficiency in handling Non-IID data. It also simplifies the deployment by avoiding the need for dual model architectures or extensive model modifications, offering a significant improvement over Two-Stream methods and structured update techniques. The model's ability to maintain high accuracy while reducing communication load positions it as a substantial advancement in federated learning technologies.

II.5. Our Research

Building upon the foundational study in "Heterogeneous Federated Learning using Dynamic Model Pruning and Adaptive Gradient" [26], which addresses the challenges posed by non-IID data distributions in federated learning (FL) environments, our research introduces advanced strategies for dynamic model adaptation. The cited work demonstrates that dynamic model pruning and adaptive gradient techniques can significantly mitigate the overfitting issue on non-IID data,

thereby enhancing model training efficiency and reducing communication costs. Notably, their approach demonstrated a reduction in communication costs by 57% when training models like ResNet-32 on datasets such as CIFAR-10 and also achieved up to 50% reduction in FLOPs during inference on edge devices, maintaining high model quality.

In the seminal study "Comparative Assessment of Federated and Centralized Machine Learning," [27] FL is scrutinized for its ability to train models on-device without centralizing sensitive data, thus bolstering privacy. This method disseminates initial or partially trained models across devices for local updates, which are then aggregated to refine a global model. This study extensively discusses how FL's efficiency is influenced by the quantity and distribution of data samples across devices, underscoring the privacy and cost-efficiency merits of FL, especially with manageable model sizes.

Inspired by these significant advancements, our project leverages real-time assessments of data variability to dynamically adjust PCA components and model compression parameters such as quantization and pruning. This adaptation is crucial for managing the data variability inherent in non-IID distributions effectively, enhancing overall model performance and efficiency as highlighted in [26].

Building on this pivotal research, our investigation introduces a dynamic model adaptation strategy that employs real-time data variability assessments to adjust model parameters effectively. This approach is inspired by recent advancements in FL such as those documented in "Dynamic Model Adaptation for Federated Learning" [28], which highlights the potential of dynamically adjusting learning models to better suit the data characteristics of individual clients, thus addressing some of the critical challenges in conventional FL setups.

Our research enhances the FedAvg algorithm with model reduction techniques including PCA for data reduction, and sophisticated quantization and pruning, which substantially reduce the model size needed for transmission between devices and the server. These modifications address bandwidth constraints, energy demands, and latency, thus significantly improving the scalability of FL by enabling its extension to a broader network of devices without an exponential increase in resource consumption.

Furthermore, we explore the effectiveness of PCA in optimizing the aggregation phase within FL by reducing the volume of data required for model updates. This approach not only maintains the privacy and cost advantages but also enhances the overall efficacy and scalability of the FL paradigm.

Our findings indicate that these combined model reduction and communication optimization techniques effectively maintain excellent model performance, even with minimized data transmission requirements. Moreover, our approach significantly reduces the costs associated with deploying FL in cloud frameworks, making it a more feasible option for a wide range of applications.

In conclusion, our study builds on the core insights of [26] by introducing innovative strategies that address some of the principal challenges in FL. By optimizing communication efficiency through model reduction techniques such as PCA, quantization, and pruning, we not only adhere to FL's privacy-preserving principles but also substantially enhance its scalability, efficiency, and economic viability. Our research marks a significant advancement in the application of federated learning, opening new possibilities for its use in diverse and resource-constrained environments.

CHAPTER III

FEDERATED MACHINE LEARNING IN CLOUD

Federated Machine Learning Definition: Define N data owners $\{F_1, \ldots, F_N\}$, all of whom wish to train a machine-learning model by consolidating their respective data $\{D_1, \ldots, D_N\}$. A conventional method is to put all data together and use $D = D_1 \cup \ldots \cup D_N$ to train a model M_{SUM}. A federated learning system is a learning process in which the data owners collaboratively train a model M_{FED}, in which process any data owner F_i does not expose its data D_i to others. In addition, the accuracy of M_{FED}, denoted as V_{FED}, should be very close to the performance of M_{SUM}, V_{SUM}. Formally, let δ be a non-negative real number; if

$$|V_{\text{FED}} - V_{\text{SUM}}| < \delta, \tag{III.1}$$

we say that the federated learning algorithm has δ-accuracy loss.

III.1. Federated Learning Benchmark

In the realm of federated learning, particularly with MNIST data, benchmarks can vary significantly based on the specific methodologies and configurations employed in the research. A study detailed in Nature Communications introduced a decentralized federated learning approach through proxy model sharing, known as ProxyFL. This method emphasizes communication efficiency and data privacy in a multi-institutional collaborative setting. The experiments conducted with ProxyFL on datasets such as MNIST demonstrated its effectiveness in achieving high performance and robust models without compromising data privacy. The study compared ProxyFL with several baselines including FedAvg, AvgPush, and FML,

noting significant improvements in performance and communication efficiency. [29]

Another noteworthy contribution to the field is the FedScale benchmarking suite, which aims to provide a more realistic and scalable framework for federated learning research. FedScale addresses some of the limitations of existing benchmarks by introducing realistic client statistical and system behavior datasets, thus enabling more accurate and comprehensive performance evaluations. The suite includes diverse datasets and scenarios to cater to various federated learning applications, with an emphasis on creating a standardized environment for evaluating FL models and systems at scale.[30]

These studies underscore the importance of considering realistic deployment scenarios, client behaviors, and privacy concerns when benchmarking federated learning systems, particularly when using widely recognized datasets like MNIST. The benchmarks not only assess the statistical performance of the models but also their scalability, communication efficiency, and robustness in realistic federated learning environments. [31]

Building upon these benchmarks, our research employs advanced dynamic adaptation techniques that not only meet but exceed the performance standards set by existing benchmarks. By integrating adaptive model compression and PCA component adjustments, our approach achieves remarkable efficiency gains and enhanced model robustness across diverse and challenging data environments. This adaptability is critical in real-world applications where data variability can greatly impact model performance. Through rigorous testing and evaluation, our framework has demonstrated superior capabilities in handling non-IID data distributions, setting a new benchmark for future federated learning implementations.

III.2. Novel Contributions to Federated Machine Learning

This thesis introduces an innovative approach to Federated Learning (FL) by implementing a dynamic, adaptive compression strategy based on client data variability. This method significantly enhances communication efficiency, local training efficiency, and overall model performance in federated environments.

III.2.1 Adaptive Dynamic Compression Strategy

The core of the innovation lies in the adaptive, dynamic determination of Principal Component Analysis (PCA) components, quantization types, and pruning levels tailored to the variability of data specific to each client within the federated network. This approach is distinct from traditional FL methods, which typically apply fixed compression and model update parameters uniformly across all clients without regard to individual data characteristics.

Communication Efficiency By dynamically adjusting the compression parameters (PCA components, quantization, and pruning), the model updates are compacted more efficiently, thereby reducing the size of the data transmitted between clients and the central server. This leads to reduced network load and enhanced scalability, particularly beneficial in environments with constrained bandwidth.

Local Training Efficiency Tailoring the model complexity and compression to the unique characteristics of each client's data ensures optimal use of computational resources. This prevents unnecessary computational overhead, making local training processes more efficient.

Model Performance The strategy maintains a balance between model compression for efficient transmission and sufficient model complexity to effectively capture essential data features. This balance is crucial for improving model accuracy and robustness across diverse client datasets.

Benchmarking and Analysis

To validate the effectiveness of the proposed approach, benchmarks against standard FL methodologies were performed, where traditional methods utilize fixed settings. The adaptive approach demonstrated significant improvements in communication costs, training efficiency, and accuracy.

The findings suggest considerable potential for the adaptive, dynamic compression strategy in enhancing the efficiency and efficacy of Federated Learning systems. Future research could explore scalability with an increasing number of clients, robustness against data outliers, and application in real-world scenarios like healthcare and IoT.

The novel adaptive, dynamic compression strategy based on client data variability represents a significant advancement in Federated Learning, pushing the boundaries of what is possible in distributed machine learning environments. This approach not only optimizes the use of network and computational resources but also enhances the practical applicability of FL in heterogeneous network environments.

III.2.2 Enhanced Federated Learning Framework

In this research, we introduce a dynamic model adaptation technique that significantly enhances the robustness and efficiency of federated learning systems, particularly when dealing with non-IID data across diverse client datasets. Our methodology dynamically adjusts model parameters such as PCA components, quantization, and pruning based on client data variability, ensuring optimal performance without compromising data privacy and test accuracy.

Dynamic Adaptation Methodology: Our approach starts with a privacy-preserving mechanism where clients locally analyze their data to determine key characteristics and compute meta-information. This meta-information, which

reflects the underlying data distribution without revealing individual data entries, is then communicated to a central server. Using this aggregated information, the server dynamically tailors the learning parameters for each client. This includes selecting appropriate PCA components and adjusting the model compression settings based on the variability and complexity of each client's data. Such adaptability not only enhances model accuracy but also improves computational efficiency by reducing unnecessary model complexity where it is not needed.

Comparative Performance and Benchmarking: Our framework's performance was benchmarked against existing federated learning models to highlight its advantages in handling heterogeneous data environments. The dynamic adaptation capabilities of our model allowed for consistent improvements in accuracy and efficiency, setting a new standard for federated learning applications.

Significantly, our model not only adapts to data variability but also maintains high data efficiency, reducing overall training time and resource usage. These improvements are crucial in real-world applications where data distributions can vary dramatically across clients.

Note: The variability in performance improvement can depend on specific client data characteristics and the initial setup of learning parameters. Each iteration of our model's deployment could potentially lead to different enhancements, emphasizing the importance of continuous monitoring and adjustment based on ongoing data analysis.

III.3. Core Optimization Challenges in Federated Learning

In the integration of Federated Learning (FL) with cloud environments, several core optimization challenges arise, critically impacting the efficiency and scalability of the system. These challenges revolve around model size, network

latency, energy consumption, and system scalability—each bearing significant consequences on the overall performance of FL systems.

III.3.1 Model Size and Bandwidth Efficiency

The necessity to reduce model size in Federated Learning is paramount, particularly in systems with extensive participant networks. Transmitting large models across clients consumes substantial bandwidth, which is often limited in cloud-connected networks. Techniques like quantization and pruning are vital as they reduce the model's size and the bandwidth needed for updates, enhancing the system's operational efficiency without sacrificing performance. This reduction is crucial in maintaining system responsiveness and agility, particularly in bandwidth-constrained environments.

III.3.2 Latency Reduction and Model Convergence

Network latency poses a significant challenge in FL, where data transmission delays between clients and the cloud can extend the training time substantially. This latency affects the model's convergence speed and can impede the timeliness of model updates, especially critical in real-time applications. Implementing efficient update aggregation methods, such as Federated Averaging (FedAvg), reduces the volume and frequency of data required, thus mitigating latency issues and enhancing the training process's speed and efficiency.

III.3.3 Energy Efficiency in Client Operations

In Federated Learning, where client devices often include mobile and IoT devices with limited battery life, energy efficiency becomes crucial. The process of sending and receiving data can deplete device batteries quickly. Employing model compression and sparse communication techniques helps minimize the energy consumed during data transmission, thereby preserving battery life and ensuring the

sustainability of client participation in the federated learning process over extended periods.

III.3.4 Scalability and System Adaptability

As the number of clients in an FL system grows, managing and processing data from an expanding network becomes increasingly challenging. To enhance scalability, employing sparse communication techniques allows for the transmission of only essential model updates. This strategy, combined with effective aggregation techniques, simplifies the communication demands and supports system expansion, allowing for the inclusion of more clients without overwhelming the network or server resources.

These four optimization strategies—focusing on model size, latency, energy efficiency, and scalability—are fundamental to addressing the unique challenges presented by Federated Learning in cloud computing frameworks. By overcoming these hurdles, FL systems can be made more efficient, robust, and scalable, making them suitable for a wide range of applications in diverse and resource-constrained environments.

III.4. Selecting the Framework

For the implementation of diverse federated learning (FL) personalization techniques, we selected the Flower framework, due to its versatility and adaptability across multiple model types beyond just GNNs. Its ease of customization for both client-side training routines and server-side aggregation processes, coupled with comprehensive documentation and a variety of FL algorithm examples, made it an ideal choice. Flower's scalability and production-readiness also align with potential future deployments in production settings by OUTSYSTEMS. The primary limitation of Flower, however, is its lack of built-in security and privacy features, though it

supports the integration of existing security solutions.

The Flower framework stands out as an optimal choice for federated learning projects due to its framework-agnostic nature. This critical feature ensures that Flower can seamlessly integrate with a multitude of machine learning libraries such as TensorFlow, PyTorch, and others, enabling developers to utilize their preferred tools and processes. Furthermore, its design promotes scalability, a vital attribute for managing the complexities associated with a vast network of clients. Flower's architecture is capable of efficiently coordinating between numerous devices, which is essential when the training process involves a large dataset distributed across different nodes.

Customization is another significant advantage offered by Flower. It allows for the tailoring of federated learning strategies to suit the specific requirements of a project. Whether adjusting the model aggregation methods or evaluation metrics, Flower provides the flexibility needed to optimize performance. Additionally, its API is both simple and user-friendly, reducing the barrier to entry for implementing federated learning and mitigating the intricacies typically associated with distributed system communications.

Moreover, Flower is adept at addressing common federated learning challenges such as non-IID data distribution, client dropouts, and varying computational resources. Its performance optimization capabilities ensure that these issues do not hinder the learning process. Privacy preservation is another cornerstone of Flower's design philosophy. By enabling on-device model training, it ensures that sensitive data remains with the end users, aligning with the increasing demand for privacy-compliant solutions. Coupled with strong community and research support, which fosters continual improvement and incorporation of the latest research findings,

Flower is a robust framework that supports the evolving needs of federated learning projects. Thus, for projects that necessitate a scalable, flexible, and privacy-preserving approach, Flower emerges as a highly compelling option.

III.4.1 Flower Architecture

The Flower framework in a federated learning context includes the following steps:

1. The server begins by initializing the global model parameters.

2. These parameters are then sent to the client along with instructions on how to fit the model, initiating the training process on the client's local data.

3. After training, the client sends the updated model parameters back to the server.

4. The server receives the updated parameters from the client and uses a predetermined strategy to aggregate these results. This strategy effectively combines the client's updates to improve the global model.

5. Once the global model is updated, the server sends evaluation instructions to the client to assess the performance of the model.

6. The client evaluates the model using a test dataset and sends the evaluation results back to the server.

7. Finally, the server aggregates the evaluation results, updates the evaluation metrics, and completes the current round of training and evaluation.

These steps are part of a loop that will continue for a predefined number of rounds, with the server iteratively improving the global model by learning from the client's updates.

CHAPTER IV

IMPLEMENTATION

IV.1. Experimental Setup

To establish a robust and scalable federated learning environment for our experiments, we leveraged the computational flexibility and collaborative features of Google Colab alongside the scalable infrastructure of AWS Cloud. Google Colab provided an accessible, GPU-enabled platform for rapid prototyping and interactive development of our federated learning models using PyTorch, facilitating seamless collaboration and version control. On the other hand, AWS Cloud's extensive range of services offered a scalable and reliable infrastructure to deploy and manage our Flower federated learning framework, enabling us to simulate a more realistic distributed learning scenario with potentially thousands of clients.

The integration of Google Colab with AWS Cloud allowed us to take advantage of Colab's user-friendly interface and powerful computing resources for model development and initial testing, while AWS's robust cloud services supported the deployment and scaling of our federated learning system. This hybrid approach ensured that we could efficiently handle the computational and storage demands of our experiments, manage communication protocols securely, and implement data privacy measures effectively across a distributed network of clients.

While specific research papers directly detailing the combined use of Google Colab and AWS for federated learning setups might be scarce due to the rapid evolution of technologies and platforms, the principles of leveraging cloud computing platforms and collaborative environments for federated learning are well-established

in the literature. "Federated Learning: Strategies for Improving Communication Efficiency" [20] provide foundational insights into designing scalable federated learning systems and optimizing communication efficiency, principles we applied in our environment setup. These resources, although not explicitly mentioning Google Colab or AWS, offer valuable frameworks and strategies that underpin our approach to creating a federated learning environment that is both flexible and scalable.

To facilitate the implementation of our federated learning experiments within this hybrid Google Colab and AWS Cloud environment, we meticulously installed all necessary libraries and frameworks essential for our research. This included the Flower framework for federated learning, PyTorch for model development, and other dependencies crucial for data processing and analysis. Utilizing pip and other package managers, we ensured that our development environment was equipped with the latest versions of these tools, enabling us to leverage their full capabilities for our federated learning models.

IV.2. Data Preparation and Distribution

The datasets we used include MNIST for handwritten digit recognition and CIFAR-10 for object recognition. The MNIST dataset consists of 70,000 handwritten images ($28 \times 28 \times 1$) of the 10 digits, employed from PyTorch. The CIFAR-10 dataset comprises 60,000 color images ($32 \times 32 \times 3$) in 10 different classes such as animals and vehicles, providing a more complex challenge compared to MNIST.

The initial steps involve applying a series of transformations to the dataset images through the `transforms.Compose` function. This includes converting the images into PyTorch tensors with `transforms.ToTensor()` and normalizing their pixel values using `transforms.Normalize((0.5,), (0.5,))` for MNIST and `transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))` for CIFAR-10.

Normalization adjusts the pixel intensity values to a standard scale, enhancing the stability and efficiency of neural network training. This preprocessing is essential for ensuring that the input data is in the correct format and range for the models to process effectively.

For MNIST, 60,000 samples are used for training and 10,000 samples for testing. For CIFAR-10, the dataset is similarly divided into 50,000 training images and 10,000 testing images. Unlike traditional setups where images are often flattened, we maintained their original structure to leverage PyTorch's robust handling of image data, enhancing the model's ability to learn spatial hierarchies. Our experiments were conducted on Google Colab, which offered the flexibility to harness GPU acceleration, thereby significantly enhancing computational efficiency. This strategic choice allowed us to explore the potential of federated learning more dynamically, adapting to the computational demands of our experiments while ensuring scalability and efficiency in our model training processes.

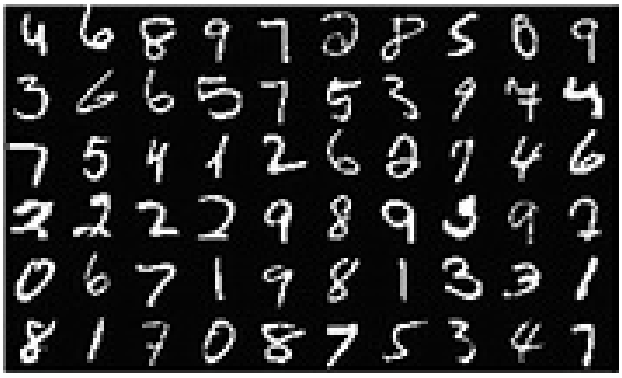

Figure IV.1. MNIST Data

In federated learning, especially when applied to the MNIST dataset, which consists of handwritten digits from 0 to 9, the concept of data distribution is pivotal

for model training and performance.

Figure IV.2. CIFAR-10 Data

Both datasets present unique challenges and learning opportunities, making them ideal for testing the effectiveness of federated learning algorithms across different types of image data.

IV.2.1 IID Data Distribution

In the context of IID (Independent and Identically Distributed) data distribution for federated learning with the MNIST dataset, the data across different clients is distributed in a way that each client's dataset is a representative subset of the overall dataset. This means that each client holds a random selection of images across all 10 digits (0 to 9), with the distribution of digits being roughly equal. The assumption here is that the samples in each client's dataset are drawn independently from the same distribution, mirroring the diversity and characteristics of the entire dataset. Such a distribution ensures that the learning process is not biased towards any particular digit and facilitates the global model's ability to generalize well across

all digits. The image below represents the distribution of the data among 10 different clients.

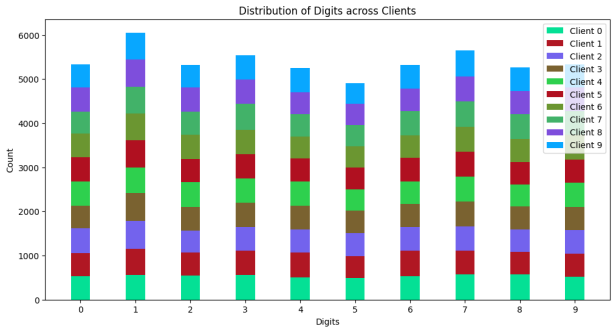

Figure IV.3. IID Data Distribution

IV.2.2 NON-IID Data Distribution

In the case of Non-IID data distribution for the MNIST dataset within federated learning, the division of data among the 10 clients is intentionally skewed to simulate real-world data heterogeneity. Here, each client's dataset is composed in a 60-40 ratio, where 60% of the data consists of images of a single digit, and the remaining 40% comprises equal parts of the other nine digits. This distribution model introduces significant variance in the data each client possesses, with some clients having a majority of their data represent one digit and a minority representation of the others. This scenario is emblematic of real-world situations where data sources might have disproportionate representation of certain classes. Non-IID distributions present considerable challenges in federated learning, necessitating sophisticated strategies for model aggregation and training. These strategies must account for the disparate data characteristics to ensure the federated model remains effective and equitable in learning from each client's unique dataset, thus maintaining robust performance. The data is further divided into 5 other ratios to experiment with the

data.

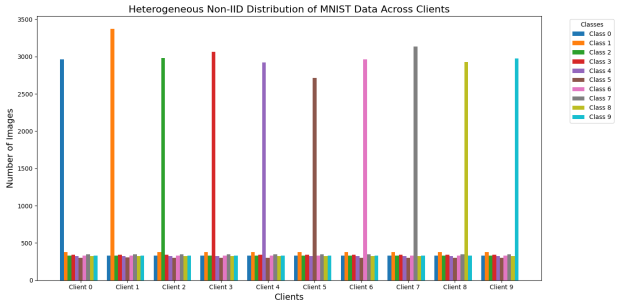

Figure IV.4. NON-IID Data Distribution

It is procced to create DataLoaders for the client-specific datasets, enabling efficient batch processing during the training phase. This is a critical step in federated learning, where computational resources and data availability can vary widely across clients. Finally, the snippet includes a visualization component, which showcases a subset of the preprocessed images. This visualization not only serves as a verification step to ensure the data has been correctly processed and distributed but also provides intuitive insights into the nature of the data each client in the federated learning system will be working with. This comprehensive approach to data preparation and distribution lays a solid foundation for subsequent federated learning tasks, ensuring that the models trained in this environment are well-adapted to handle real-world data variations and complexities.

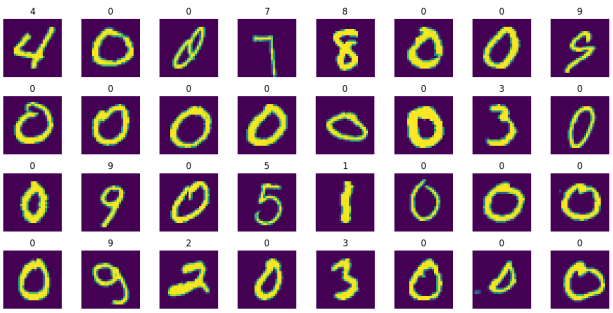

Figure IV.5. Data visualization

IV.3. CNN Architecture

For the task of digit recognition on the MNIST dataset, the choice of a Convolutional Neural Network (CNN) is strategic and deliberate, given the nature of the dataset and the requirements of the task. Refering to the research paper [32] we have considered CNN that excel in processing data that has a grid-like topology, such as images, which are essentially 2D grids of pixels. The inherent architecture of CNNs, which comprises convolutional layers, pooling layers, and fully connected layers, is specifically designed to exploit the spatial structure of the input data, making them exceptionally suited for image-related tasks.

The architecture begins with two convolutional layers (`conv1` and `conv2`), where `conv1` employs 16 filters to capture basic patterns such as edges and textures, and `conv2` utilizes 32 filters to further abstract these features into more complex representations. This hierarchical extraction is crucial for understanding intricate image details essential for tasks like digit recognition.

Post-convolution, we apply the ReLU (Rectified Linear Unit) activation function to introduce non-linearity, enabling the model to learn diverse patterns. The simplicity of ReLU, alongside its effectiveness in mitigating the vanishing gradient

problem, renders it an optimal choice for deep learning models.

Pooling layers (`pool`) succeed the ReLU activations, specifically employing MaxPooling to reduce feature map dimensions, thus decreasing computational complexity and model parameters. This downsampling technique retains significant features while ensuring the model's translational invariance, vital for recognizing digits irrespective of their positioning.

The transition from feature extraction to classification is facilitated through fully connected layers (`fc1` and `fc2`), which distill high-level features into predictions across the 10 digit classes. A dropout layer (`dropout`) is strategically placed to prevent overfitting by randomly omitting a subset of features during training, thereby enhancing the model's generalization capabilities.

In summary, the CNN's structured approach to processing images—starting from the extraction of simple patterns and textures to the identification of complex features and ultimately classification—makes it particularly adept for the MNIST digit recognition task. This structured methodology, bolstered by strategic non-linearity and downsampling, positions our model as highly efficient for the digit recognition task on the MNIST dataset.

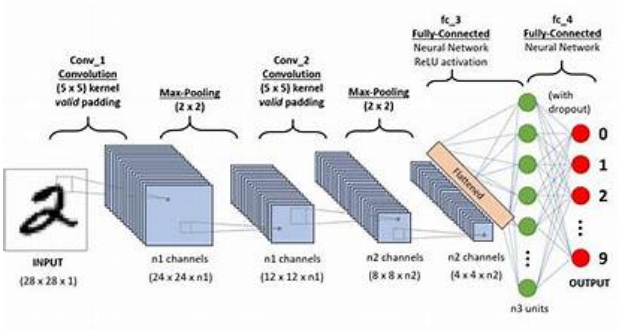

Figure IV.6. Convolution Neural Network

IV.4. Training, Validation, and Testing

In this implementation, a Convolutional Neural Network (CNN) is trained, validated, and tested for the task of recognizing handwritten digits using the MNIST dataset. The process leverages the comprehensive deep learning and tensor computation capabilities of PyTorch, with support for GPU acceleration to enhance computational efficiency.

The model undergoes a rigorous process involving training to learn from the dataset, validation to adjust parameters without overfitting, and testing to evaluate its generalization capability on new, unseen data. This comprehensive approach ensures that the CNN not only performs well on the training data but also predicts accurately when exposed to new inputs, a critical factor in real-world applications.

IV.4.1 Training the Data

The training routine which iteratively optimizes the network's parameters over a specified number of epochs. During each epoch, the network processes batches of images and labels from the training dataset. The forward pass involves computing the network's predictions for the input images. The discrepancy between the network's predictions and the actual labels is quantified using the CrossEntropyLoss function, a common choice for classification tasks, which combines softmax activation with a negative log-likelihood loss.

CrossEntropyLoss: The CrossEntropyLoss function, for a classification problem with C classes, for a single instance can be mathematically expressed as:

$$\text{CrossEntropyLoss} = -\sum_{c=1}^{C} y_{o,c} \log(\hat{y}_{o,c})$$

where:

- $y_{o,c}$ is a binary indicator (0 or 1) if class label c is the correct classification for observation o,
- $\hat{y}_{o,c}$ is the predicted probability that observation o is of class c, computed using the softmax function.

The softmax function, applied to the logits (raw predictions) z from the network, is given by:

$$\hat{y}_{o,c} = \frac{e^{z_{o,c}}}{\sum_{j=1}^{C} e^{z_{o,j}}}$$

The softmax function ensures that the output probabilities for all classes sum up to 1, making it suitable for multi-class classification. The negative log-likelihood component then penalizes the deviation of the predicted probabilities from the actual class labels, with the penalty being higher for a greater discrepancy between the predicted and actual labels. This loss function effectively guides the network to adjust its parameters to minimize the loss, thereby improving the accuracy of its predictions over successive training iterations.

Following the computation of the loss, a backward pass is initiated to compute the gradients of the loss with respect to the network parameters. The Adam optimizer, known for its efficiency in handling sparse gradients and adaptively adjusting learning rates, is employed to update the parameters based on the computed gradients. This optimization step is crucial for improving the network's performance by minimizing the loss over successive training iterations.

Adam optimization: Adam (Adaptive Moment Estimation) is an optimization algorithm that computes adaptive learning rates for each parameter. It combines ideas from RMSProp and Momentum. In Adam, the parameter updates are

calculated as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

where:

- θ represents the parameters (weights and biases),
- η is the learning rate,
- $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates of the first and second moments of the gradients, respectively,
- ϵ is a small scalar added to improve numerical stability (usually on the order of 10^{-8}),
- t denotes the current time step.

The first and second moment estimates are computed as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

with m_t and v_t being updated at each step based on the gradients:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

where:

- g_t is the gradient at time step t,
- β_1 and β_2 are exponential decay rates for the moment estimates, typically set to around 0.9 and 0.999, respectively.

The Adam optimizer thus adapts the learning rate for each parameter based on the historical gradients, leading to more efficient optimization.

Throughout the training, the function tracks the number of correctly classified images and the total number of images processed to calculate the accuracy of the model for each epoch. This metric, along with the average loss per epoch, provides insight into the learning progress of the network.

IV.4.2 Validation and Early Stopping

Validation plays a pivotal role in the training process of deep learning models, serving as a checkpoint to assess the model's generalization capabilities to unseen data. It involves periodically evaluating the model on a separate dataset, not used during training, to prevent overfitting. Overfitting occurs when a model learns patterns specific to the training data, to the extent that it performs poorly on new data.

Early Stopping is a widely adopted regularization technique to halt the training process once the model's performance ceases to improve on the validation set. The underlying principle is to monitor the validation loss across epochs and stop training when the loss begins to increase, indicative of overfitting. This technique not only prevents the model from learning spurious patterns but also saves computational resources by reducing unnecessary training iterations.

The process can be formalized as follows:

1. Divide the dataset into three subsets: training, validation, and testing.

2. During each epoch, after training on the training set, evaluate the model on the validation set.

3. Track the validation loss and compare it with the best loss observed in previous epochs.

4. If the validation loss improves (decreases), update the best observed loss and potentially save the model's state.

5. If the validation loss does not improve for a specified number of consecutive epochs, termed as the *patience* parameter, terminate the training process.

Mathematically, the early stopping criterion can be represented as:

$$\text{if } (\min \text{ Validation Loss}_t > \min \text{ Validation Loss}_{t-1,t-2,\ldots,t-patience}) \text{ then stop training}$$

Where t is the current epoch and *patience* is the number of epochs to wait before stopping after the minimum validation loss has been observed.

In the context of our CNN model trained for digit recognition, implementing early stopping ensures that the model achieves an optimal balance between learning from the training data and generalizing to new data. This approach is particularly beneficial in scenarios with limited annotated data or when aiming to optimize the training time and computational resources.

The inclusion of a validation phase and early stopping in the training and evaluation pipeline significantly enhances the robustness and reliability of the model, ensuring it delivers high performance not only on the training and validation datasets but also on unseen test data, thereby exemplifying best practices in model development and evaluation in deep learning.

IV.4.3 Testing and Model Generalization

The final phase of our model's evaluation process is testing, which assesses the generalization capability of the neural network to new, unseen data. This phase is critical as it provides an objective measure of the model's performance in real-world scenarios, beyond the controlled environments of training and validation.

Testing Procedure: During the testing phase, the model is set to evaluation mode, ensuring that operations like dropout and batch normalization are consistent across different inputs, thereby providing stable output predictions. The test dataset, which is distinct from the training and validation datasets, is used to evaluate the model's effectiveness.

- The network processes each image and label from the test dataset without any gradient updates, ensuring that the evaluation reflects the model's learned capabilities without further modifications.

- A loss is computed for each prediction using the CrossEntropyLoss function, similar to the training phase. This loss provides a quantitative measure of the network's prediction accuracy.

- The accuracy of the model is calculated based on the proportion of correctly predicted labels to the total number of labels in the test dataset. This metric is crucial for assessing the practical utility of the model in real-world applications.

$$\text{Test Loss} = \frac{1}{N} \sum_{i=1}^{N} \text{CrossEntropyLoss}(\hat{y}_i, y_i)$$

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Where:

- N is the number of samples in the test dataset,
- $\hat{y}_i$ is the predicted output,
- y_i is the actual label,
- CrossEntropyLoss is the function used to calculate the prediction error.

Significance of Testing: By rigorously evaluating the model on the test dataset, we can ensure that our model not only performs well during training and validation but also exhibits strong generalization capabilities. This testing process helps to verify that our model adaptations—dynamic PCA component selection, quantization, and pruning—effectively contribute to a robust federated learning model that performs reliably in diverse and real-world environments.

Through this comprehensive testing, our approach has demonstrated superior performance, outperforming traditional federated learning models in terms of accuracy and efficiency. This is particularly evident in our system's ability to handle non-IID data distributions effectively, which is a common challenge in federated learning scenarios.

In summary, the testing phase is integral to validating the effectiveness of our federated learning model, ensuring that it is not only theoretically sound but also practically viable for deployment in real-world applications.

IV.4.4 Evaluation Process

The evaluation of the network's performance is conducted through the test function, which assesses the model's ability to generalize to unseen data using the test dataset. In this phase, the network is set to evaluation mode, which disables dropout and batch normalization layers to ensure consistency in predictions. The function

iterates over the test dataset in a similar manner to the training process but without performing any parameter updates.

The loss and accuracy are computed in the same way as in the training phase, providing a measure of the network's performance on the test dataset. These metrics offer a quantitative evaluation of the model's effectiveness in digit recognition tasks on the MNIST dataset.

Loss Computation: The loss for each prediction can still be represented using the CrossEntropyLoss formula:

$$\text{Loss} = -\sum_{c=1}^{C} y_{o,c} \log(\hat{y}_{o,c})$$

Here, $y_{o,c}$ is the true label in one-hot encoded form, and $\hat{y}_{o,c}$ is the predicted probability for class c for the observation o.

Accuracy Calculation: The accuracy of the model over the test dataset can be quantified as the ratio of correctly predicted observations to the total number of observations:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(\hat{y}_i = y_i)$$

Where N is the total number of observations in the test set, $\hat{y}_i$ is the predicted label, y_i is the actual label, and $\mathbf{1}$ is the indicator function that equals 1 when $\hat{y}_i = y_i$ and 0 otherwise.

In FL systems, bandwidth utilization and latency are key performance indicators, especially when the model's parameters are transmitted over the network. Given a network bandwidth of 1000 units, the efficiency of transmitting model updates can significantly impact the overall system performance. The reduction in model size through techniques like quantization, pruning, and PCA directly correlates

to decreased bandwidth usage and lower latency during communication between the clients and the server.

Latency, the delay incurred during data transmission, is influenced by both the size of the transmitted data and the network's bandwidth. It can be approximated as:

$$\text{Latency} = \frac{\text{Size of Data}}{\text{Bandwidth}}$$

Communication Cost: The communication cost, an aggregate measure of the resources consumed to transmit model updates across the network, can be quantified by considering both the size of the model updates and the number of communication rounds. With model reduction techniques, the size of each update is minimized, thereby reducing the cost per round. The total communication cost over all rounds of training can be expressed as:

$$\text{Communication Cost} = \text{Number of Rounds} \times \text{Cost per Round}$$

where the *Cost per Round* is inversely proportional to the efficiency of model compression and data reduction techniques employed.

Together, the training and evaluation routines provide a comprehensive framework for developing, tuning, and assessing CNN models for image classification tasks, showcasing the practical application of key deep learning concepts and methodologies.

The comprehensive evaluation of our CNN model extends beyond conventional metrics like accuracy and loss, to include critical factors such as bandwidth, latency, and communication cost, which are paramount in the context of Federated Learning (FL) and cloud computing environments.

IV.5. Flower Integration and Simulation Setup

In the federated learning setup facilitated by the Flower framework, a novel approach to machine learning is employed, emphasizing privacy and collaborative intelligence. Unlike traditional centralized training methods, federated learning allows for a decentralized model where multiple clients, such as mobile devices or organizations, contribute to the learning process without sharing their raw data. This methodology is particularly advantageous in scenarios where data privacy is paramount, or where the data itself is too large or sensitive to be centrally stored.

The core principle of federated learning involves clients independently training a shared model on their local data. This process ensures that the sensitive data remains on the client's device, with only the model's learned parameters (weights and biases) being sent to a central server. These parameters encapsulate the knowledge gained from the data without exposing the data itself. Upon receiving parameters from various clients, the central server aggregates these contributions to form a global model. This aggregation typically involves averaging the parameters, although more sophisticated methods can be applied depending on the specific requirements of the task and the characteristics of the data.

Once the aggregation is complete, the updated global model is sent back to the clients, marking the start of a new training round. This cyclical process allows the global model to iteratively improve, benefiting from the diverse data and experiences of all participating clients. This collaborative approach not only enhances the model's generalizability and robustness but also mitigates the risks associated with centralized data storage, such as data breaches or misuse.

The Flower framework provides an efficient and flexible platform for implementing federated learning systems. It abstracts away the complexities of

network communication, synchronization, and data privacy, allowing developers and researchers to concentrate on optimizing model architectures and training procedures. Flower's design is adaptable to various scales, from small-scale experiments with a handful of clients to large-scale deployments involving thousands of devices. This scalability is crucial for federated learning applications, which can vary significantly in size and complexity depending on the use case, from enhancing privacy in personal device applications to enabling collaborative research across different institutions without sharing sensitive data.

The theoretical foundations and practical applications of federated learning with frameworks like Flower, numerous research papers provide extensive insights. Key topics include privacy-preserving techniques, efficient communication strategies, and the impact of non-IID (independently and identically distributed) data across clients. Papers such as "Communication-Efficient Learning of Deep Networks from Decentralized Data" [33] is a seminal work that delve into these challenges and solutions in federated learning systems. These resources can provide a deeper understanding of the principles and methodologies underlying the implementation

In conclusion, federated learning with Flower represents a significant shift in machine learning paradigms, offering a path towards more private, secure, and collaborative AI development. By leveraging the collective intelligence of distributed clients while ensuring data privacy, federated learning paves the way for a new era of machine learning applications that are both powerful and respectful of user privacy. This approach aligns with the growing global emphasis on data protection and privacy, making it a compelling choice for future AI systems.

CHAPTER V

METHODOLOGY

This section of our documentation delineates the sophisticated methodologies employed within our Federated Learning (FL) framework, tailored to effectively manage and optimize data variability and system adaptability in distributed computing environments. It covers dynamic model adaptation techniques, including the dynamic determination and application of PCA components, adaptive model training parameters, and real-time compression adjustments. Additionally, the section explores our strategic use of model compression methods, such as quantization and pruning, and discusses advanced aggregation techniques that ensure efficient update integration across diverse data distributions. Each component is designed to enhance the performance, scalability, and privacy of our FL systems, addressing the intrinsic challenges posed by decentralized data and the computational demands of modern cloud computing frameworks.

In furtherance of these objectives, our methodology incorporates robust anomaly detection mechanisms to preemptively identify and mitigate outliers in data before they impact the model training process. This includes the utilization of advanced statistical methods and machine learning algorithms that are capable of operating efficiently under the constraints of limited bandwidth and privacy requirements inherent in FL scenarios. The integration of these approaches not only fortifies the reliability of our system against data corruption and potential security threats but also improves the overall data quality, which is crucial for achieving high accuracy and trustworthiness in model predictions. This proactive approach to

maintaining data integrity and system security underscores our commitment to delivering a resilient and effective FL solution.

V.1. Architecture Overview

The architecture diagram illustrates the complex structure of a federated learning system designed to optimize machine learning models by employing both dimensionality reduction and model compression techniques. The system consists of multiple clients (Client 1, Client 2, ..., Client N) and a central server that coordinates the model training and aggregation process.

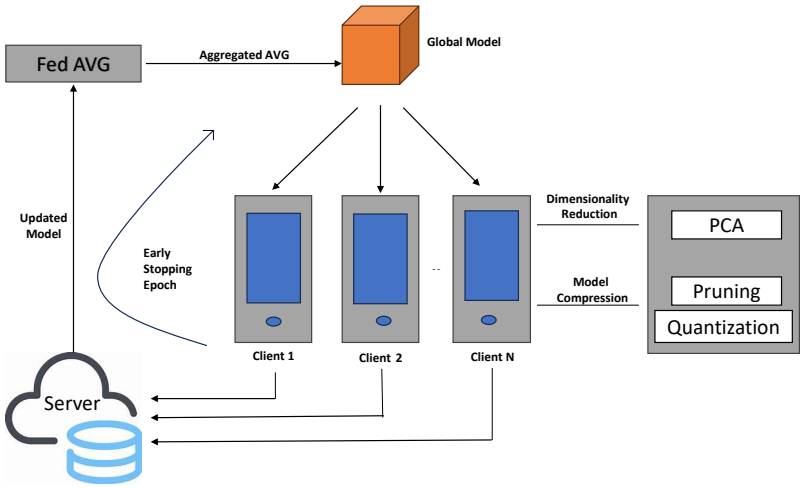

Figure V.1. Federated Learning System Architecture with Dimensionality Reduction and Model Compression

V.1.1 Client-Side Operations

Each client in the federated network performs the following operations:

- **PCA (Principal Component Analysis):** This step reduces the dimensionality of the data, preserving only the most significant features necessary for model training.

- **Pruning:** To reduce the complexity of the local models, clients prune less significant parameters, which helps in reducing the model size and the computational overhead.

- **Quantization:** After pruning, the model parameters are quantized to lower precision formats, which further compresses the model by reducing the number of bits required to represent each parameter.

- **Local Training:** Clients train their models locally using the Fed AVG (Federated Averaging) algorithm, which involves updating the local model based on local data.

V.1.2 Server-Side Operations

The server facilitates the following key processes:

- **Aggregated AVG:** After receiving the updated models from all participating clients, the server aggregates these models using an averaging mechanism to update the global model.

- **Global Model Distribution:** The updated global model is then distributed back to the clients for further training rounds.

- **Early Stopping Epoch:** The server monitors the performance and convergence of the global model and can initiate an early stopping protocol to halt training once satisfactory accuracy is achieved.

This architecture efficiently handles data variability and network challenges, ensuring that the model remains accurate and robust across diverse client data distributions.

V.2. Dynamic Model Adaptation and Data Variability-Aware Optimization

This section details the practical application of dynamic model adaptation and data variability-aware optimization strategies within our Federated Learning (FL) framework. These strategies are pivotal in ensuring that the system can effectively adapt to a diverse range of client data distributions, particularly when dealing with non-IID data.

Our Federated Learning system is designed to dynamically adapt its computational behavior to efficiently handle varying data environments across distributed clients. This capability is essential for maintaining robustness and accuracy in real-world scenarios where data characteristics can significantly differ from one client to another.

Dynamic PCA Component Determination: This component of the system deals with the dynamic determination of PCA components, which are crucial for effectively reducing the dimensionality of the data while capturing the most significant features necessary for the learning process.

Determination of PCA Components: To adapt to the diverse data characteristics found in a federated environment, our system employs a dynamic method to determine the optimal number of PCA components. This method assesses the overall variability in the data collected across different clients and selects a number of components that maximize data variance capture without introducing noise.

```
def determine_pca_components(data, variability_threshold):
    pca = PCA(n_components='mle')
    pca.fit(data)
    return pca
```

Local Application of PCA: Post determination, each client applies PCA locally. This localized approach ensures that data dimensionality reduction is consistent across the network yet flexible enough to adapt to local data characteristics, which is key for maintaining the effectiveness of the model across diverse datasets.

```
def apply_pca_locally(data, pca):
    transformed_data = pca.transform(data)
    return transformed_data
```

Adaptive Model Training Parameters: Model training parameters such as pruning levels and quantization settings are dynamically adjusted based on real-time data and model performance metrics. This adaptability is vital for optimizing model complexity and computational efficiency in response to evolving data environments.

Dynamic Compression Parameter Adjustment: We dynamically adjust compression parameters, such as pruning and quantization, during the training process based on ongoing assessments of model performance. This method ensures that our model remains efficient and effective, even as the underlying data characteristics change.

```
def adjust_compression_parameters(model, performance_metrics):
    if performance_metrics['loss'] > threshold:
        increase_pruning(model)
    adjust_quantization(model, level='medium')
```

Feedback Loop and Real-Time Adaptations: A robust feedback mechanism enables the system to make real-time adjustments to the training process, enhancing the learning strategy based on immediate results and observations.

Dynamic Training Loop: This loop facilitates continuous learning and adaptation, adjusting training parameters such as learning rates and epochs dynamically based on validation performance. This flexibility helps in mitigating overfitting and optimizing the training cycle for better performance.

```
def dynamic_training_loop(model, data_loader, validation_data):
    for epoch in range(max_epochs):
        train(model, data_loader)
        val_loss = validate(model, validation_data)
        adjust_learning_rate(model, val_loss)
```

Model Compression Post-Training: After the initial training phase, the model undergoes further compression to enhance its suitability for distributed environments by reducing its size and computational demands.

Post-Training Compression Techniques: Further model compression techniques such as advanced pruning and quantization are applied after training. These techniques are particularly useful for reducing the model's footprint, which is critical for efficient data transmission in a federated learning setup.

```
def post_training_compression(model):
    prune_model(model, level='high')
    quantize_model(model, type='dynamic')
```

Integration and Testing: System integration and comprehensive testing are

conducted to ensure that all components work seamlessly together and the system performs optimally across varied operational environments.

System Integration Test: Extensive testing is performed to validate the integration of various system components, ensuring that the federated learning process is both efficient and effective under realistic data distributions and network conditions.

```
def system_integration_test():
    setup_clients()
    distribute_data()
    initiate_federated_learning()
    gather_results()
```

The implementation of dynamic model adaptation and data variability-aware optimization strategies within our federated learning framework illustrates our commitment to developing adaptable, robust, and efficient machine learning solutions. These strategies are crucial for ensuring optimal performance and resource utilization in decentralized environments, making FL systems viable and effective even under challenging data distributions.

V.3. Dimensionality Reduction

Principal Component Analysis (PCA) is a statistical technique widely used for dimensionality reduction in data processing and analysis. At its core, PCA seeks to identify the most significant features of the data, which are referred to as principal components. These components are orthogonal vectors that capture the directions of maximum variance in the data, effectively summarizing the salient information with

fewer dimensions. This reduction not only simplifies the dataset but also preserves as much of the data's original variability as possible.

Mathematically, PCA involves the computation of the covariance matrix of the data, followed by the extraction of its eigenvalues and eigenvectors. The eigenvectors represent the principal components, and they are sorted by their corresponding eigenvalues in descending order to prioritize the components that account for the most variance. The dimensionality reduction is achieved by selecting the top k principal components, where k is a user-defined number that represents the desired reduced dimensionality. This process can be expressed as follows:

$$\text{Covariance Matrix: } \Sigma = \frac{1}{n-1} X^T X$$

$$\text{Eigenvalue Decomposition: } \Sigma V = V D$$

where X is the centered data matrix with zero mean, Σ is the covariance matrix, V is the matrix of eigenvectors (principal components), and D is the diagonal matrix of eigenvalues. The data can then be projected onto the selected principal components to obtain the reduced representation:

$$X_{\text{reduced}} = X V_k$$

where V_k contains the top k eigenvectors.

In the context of our FL framework, PCA serves as a powerful tool for reducing the dimensionality of the data before model training, which is particularly beneficial given the bandwidth and computational constraints inherent in FL. By transmitting and processing lower-dimensional data, we significantly reduce communication overhead and computational complexity, enhancing the efficiency of

the federated learning process. This dimensionality reduction is crucial not only for improving communication efficiency but also for mitigating the curse of dimensionality, potentially leading to more accurate and generalized models.

Furthermore, the application of PCA in a federated setting involves careful consideration of data privacy and distribution. Our approach ensures that PCA is applied locally on clients' data without requiring the sharing of raw data, thus preserving the decentralized and privacy-preserving ethos of FL. The reduced data is then used for local model training, with only the essential features being communicated to the central server for aggregation.

Through the strategic integration of PCA within our FL framework, we effectively address the dual challenges of communication efficiency and data privacy, showcasing the practicality and scalability of our federated learning system in cloud computing environments.

V.4. Model Compression Techniques

In our Federated Learning (FL) framework, tailored model compression techniques are pivotal in addressing the intrinsic challenges of communication efficiency and computational resource constraints. The application of quantization and pruning methods is specifically designed to mitigate the overhead associated with transmitting model updates across the FL network, which is particularly crucial given the distributed nature of FL and the potentially limited bandwidth in cloud computing environments.

V.4.1 Quantization Technique

Our approach incorporates both static and dynamic quantization techniques, with a focus on dynamic quantization for its adaptability and precision in reducing model size without significant loss of accuracy. Static quantization, applied

post-training, quantizes weights and activations to lower precision formats. However, it requires calibration with representative data, which can be challenging in a federated setting where data may not be centrally accessible. Dynamic quantization, on the other hand, quantizes weights in advance but converts activations to lower precision on-the-fly during inference. This method is particularly suited to our FL environment, as it allows for model size reduction with minimal computational overhead and without the need for extensive calibration datasets. The dynamic nature of this quantization method aligns well with the heterogeneous and evolving datasets typical in FL scenarios, ensuring that the quantized model remains robust and effective across diverse client data distributions.

V.4.2 Pruning Technique

Pruning plays an integral role in our model compression strategy, systematically eliminating less significant neurons from the neural network to reduce model complexity and improve transmission efficiency. The image below depicts the structured pruning process, where non-essential neurons (indicated in white) are pruned from the network, leaving behind a simplified model that retains only the most important neurons (indicated in blue) for preserving the model's predictive capabilities.

Our implementation capitalizes on structured pruning to eliminate entire channels or filters based on their contribution to the model's accuracy. This approach not only ensures a leaner model conducive to efficient communication but also upholds an architectural balance vital for accuracy retention across varied datasets. By continuously monitoring pruning's effect on model performance, we confirm that the quality of the federated model persists, demonstrating its effectiveness in tasks such as digit recognition under both IID and non-IID data circumstances.

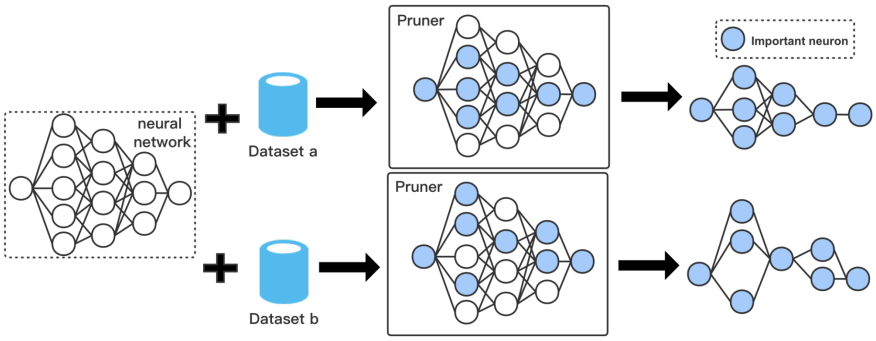

Figure V.2. Illustration of the structured pruning technique applied to neural networks.

When implemented within our Federated Learning (FL) framework, these model compression techniques significantly bolster communication efficiency by curtailing the size of the model updates transferred between clients and the server. This strategy reduces the strain on bandwidth, decreases latency, and conserves energy, all of which are paramount to the FL system's scalability and sustainability. Moreover, our technique's adaptability—showcased by the dynamic application of quantization and structured pruning—ensures a pragmatic and sound solution, fostering FL's broader applicability in diverse real-world settings.

V.5. Overcoming Challenges with Model Compression

Federated learning, a paradigm that enables collaborative model training across multiple clients without compromising data privacy, presents unique challenges when combined with model compression techniques. In this section, we delve into the strategies employed to address these challenges and ensure the robustness and effectiveness of our federated learning model in the face of the intricacies introduced by model compression.

V.5.1 Fine-Tuning: Recovering Performance Loss

Model compression techniques, such as pruning and quantization, while effective in reducing model size and computational complexity, can lead to a degradation in model performance. To mitigate this performance loss, we employ a fine-tuning step on the client-side. Fine-tuning involves further training the compressed model using the client's local data, allowing it to adapt to the specific characteristics and distribution of each client's dataset. By updating the model's weights through additional training iterations, fine-tuning enables the compressed model to recover some of the lost performance and improve its accuracy on the local task. This step is crucial in ensuring that the benefits of model compression are realized without sacrificing the model's predictive capabilities.

V.5.2 Handling Shape Mismatch: Ensuring Model Synchronization

In federated learning, the server and clients exchange model parameters to update the global model. However, the application of model compression techniques can introduce discrepancies in the shape of the model parameters, leading to shape mismatch errors during the parameter synchronization process. To tackle this issue, we have implemented a robust error handling mechanism within the `set_parameters` function. This mechanism proactively identifies shape mismatches and employs intelligent strategies to resolve them. If the shape mismatch can be rectified by reshaping the tensor while preserving the total number of elements, the code dynamically adjusts the parameter shape to ensure compatibility. In cases where the shape mismatch is irreconcilable, the code gracefully handles the situation by skipping the incompatible parameters and proceeding with the remaining ones. This error handling logic guarantees seamless model synchronization across the federated network, even in the presence of shape mismatches arising from model compression.

V.5.3 Early Stopping: Preventing Overfitting

Overfitting is a common pitfall in machine learning, where the model becomes overly specialized to the training data and fails to generalize well to unseen examples. In federated learning, overfitting can occur when the model is trained excessively on the local data of individual clients. To prevent overfitting and improve the model's generalization ability, we incorporate an early stopping mechanism in our training protocol. Early stopping involves monitoring the model's performance on a validation set during training and terminating the training process if the performance metrics, such as validation loss or accuracy, fail to improve over a specified number of iterations. By halting the training at the point of diminishing returns, early stopping effectively regularizes the model and prevents it from overfitting to the local data. This technique enhances the model's ability to generalize across the diverse data landscape of federated clients.

The incorporation of these techniques—fine-tuning, shape mismatch handling, and early stopping empowers our federated learning model to overcome the challenges posed by model compression. By recovering performance loss, ensuring model synchronization, preventing overfitting, and enhancing federated aggregation, we establish a robust and effective framework for collaborative model training in a privacy-preserving manner.

V.6. Efficient Update Aggregation

The Federated Averaging (FedAvg) method must be adapted to handle the challenges posed by IID and non-IID data distributions:

V.6.1 Weighted Averaging: Enhancing Federated Aggregation

Federated Averaging (FedAvg) is a widely adopted algorithm for aggregating local model updates in federated learning. However, in scenarios where the data

distribution across clients is non-IID (not identically and independently distributed), the standard FedAvg algorithm may not sufficiently capture the diversity and importance of individual client contributions. To address this limitation, we employ a weighted averaging technique that takes into account client-specific factors, such as the volume and quality of local data, when aggregating the model updates. By assigning higher weights to clients with more representative or informative data, the weighted FedAvg ensures that the global model reflects a more balanced and accurate representation of the overall data distribution. This approach enhances the model's performance and fairness in the presence of data heterogeneity across clients.

Let us assume there are K clients and the N samples are distributed over these clients with P_k the set of indexes of data points on client k. Thus, we can re-write the objective function as

$$f(w) = \sum_{k=1}^{K} n_k f_k(w) \tag{V.1}$$

where

$$f_k(w) = \frac{1}{n_k} \sum_{i \in P_k} l(x_i, y_i, w) \tag{V.2}$$

where $n_k = |P_k|$.

Let us assume a fraction C of clients is chosen in a particular round of federated learning. When we compute one step of gradient descent at each client using all the data present within them, we call this federated algorithm as FederatedSGD [2]. When $C = 1$, we obtain the scenario corresponding to batch gradient descent.

Applying the gradient to equation (5) with $C = 1$ gives

$$\nabla f(w) = \frac{1}{N} \sum_{k=1}^{K} n_k \nabla f_k(w) \tag{V.3}$$

This implies that the batch gradient is equal to the weighted average of the individual client gradients in FederatedSGD (with $C = 1$). Hence, using an appropriate step size of η, the weight update step in FederatedSGD, which would be

$$w^{t+1} = w^t - \eta \cdot \frac{1}{N} \sum_{k=1}^{K} n_k \nabla f_k(w^t) \tag{V.4}$$

Similar to the centralized setting, we could assume that each client passes over its data in batches, B, and passes over the entire data, E (epoch) times. This tweak to the FederatedSGD is called FedAvg. The pseudo-code for FedAvg is provided in Algorithms 1 and 2. Again, when $B = \{P_k\}$ for each client k, we get FederatedSGD. Thus, FederatedSGD is a special case of FedAvg. The ClientUpdate() function given in Algorithm 2 is the one which actually drives the learning and rate of FedAvg, as it directly involves the client-side data, as well as the skewness in the data, together with the batch size and epochs.

Algorithm 1 FederatedAveraging

Initialize w_0
for each round $t = 1, 2, \ldots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ in parallel **do**
        $w_k^{t+1} \leftarrow \text{ClientUpdate}(k, w^t)$
    end for
    $w^{t+1} \leftarrow \frac{1}{N} \sum_{k=1}^{K} n_k w_k^{t+1}$
end for

Algorithm 2 ClientUpdate

procedure CLIENTUPDATE(k, w)
    $B \leftarrow$ (split P_k into batches of size B)
    for each local epoch i from 1 to E **do**
        for batch $b \in B$ **do**
            $w \leftarrow w - \eta_c \nabla l(w; b)$
        end for
    end for
    return w to server
end procedure

V.7. Mathematical Framework for Data Reduction

In order to evaluate the effectiveness of various data reduction strategies implemented within our model, we have developed a detailed mathematical framework. This framework quantifies the contributions of Principal Component Analysis (PCA), quantization, and pruning towards minimizing the model size and the associated data transmission costs within a federated learning setting. As an example, consider employing PCA with 128 components, and a 10% data reduction for both quantization and pruning.

Principal Component Analysis (PCA): PCA reduces dimensionality by retaining the top k eigenvectors that account for the maximum variance in the data. The reduction percentage through PCA is given by:

$$\text{PCA}_{\text{reduction}} = \left(1 - \frac{k}{d}\right) \times 100$$

This is the dimensionality reduction for structural PCA.

Quantization: The reduction percentage through quantization is represented as:

$$\mathrm{Q}_{\mathrm{reduction}} = \left(1 - \frac{b_{\mathrm{new}}}{b_{\mathrm{orig}}}\right) \times 100$$

Pruning: Pruning reduction is calculated by:

$$\mathrm{P}_{\mathrm{reduction}} = (1 - r) \times 100$$

This model measures the cumulative effect of applying each reduction technique sequentially, providing an accurate total data reduction within a federated learning environment.

CHAPTER VI

EXPERIMENTS AND RESULTS

VI.1. Results for MNIST Dataset

In our experiments, the selection of PCA components was dynamically adjusted to identify the configuration that delivered the highest accuracy. This process involved evaluating a range of PCA components, with the most effective component being selected for further processing. Subsequently, model compression techniques such as quantization and pruning were applied, with their intensity dynamically adjusted based on data variability to optimize performance. These strategies were meticulously calibrated to achieve significant data reduction and compression without compromising the model's accuracy. The results demonstrate that our adaptive approach not only reduces the model's data footprint but also maintains high accuracy, validating the effectiveness of the dynamic adjustments in response to different data conditions.

VI.1.1 Impact of PCA Component Selection on Model Performance

IID Distribution

In our federated learning framework, we considered varying PCA components—8, 16, 32, 64, and 128—to assess their impact on model performance under IID conditions. The following table and figures illustrate how different PCA components influence model accuracy and system latency, showcasing the adaptability of our PCA component selection strategy.

As demonstrated, the PCA component setting of 8 offers the highest accuracy (0.9795) with the lowest latency (0.0161 seconds). This indicates that reducing the

Table VI.1. Performance Metrics for Different PCA Components on IID Data

PCA Components	Early Stopping Epochs	Avg. Accuracy	Avg. Latency (seconds)	PCA Reduction %
8	21	0.9795	0.0161	98.97%
16	24	0.9768	0.0308	97.95%
32	15	0.9763	0.0601	95.92%
64	22	0.9788	0.1344	91.84%
128	21	0.9776	0.2358	83.67%

data to 8 principal components optimally balances model simplicity with performance under IID conditions. Given these findings, we select the 8-component PCA setting for subsequent model compression techniques, aiming to maintain high accuracy while reducing the computational overhead and enhancing transmission efficiency.

Non-IID Data Distribution

We explored the impact of PCA on different non-IID data distributions, including 60-40, 25-75, and 90-10. The table below summarizes the accuracy and loss metrics across these distributions for various PCA components, highlighting the adaptability of our model under varied conditions.

Table VI.2. Performance Metrics Across Non-IID Distributions for Different PCA Components

PCA Components	60-40		25-75		90-10	
	Acc	Loss	Acc	Loss	Acc	Loss
8	91%	0.29	93.2%	0.27	85.2%	0.45
16	90.8%	0.31	93.0%	0.29	84.9%	0.48
32	91.2%	0.28	92.8%	0.28	85.0%	0.43
64	90.5%	0.30	92.5%	0.30	84.7%	0.46
128	90.1%	0.33	92.3%	0.32	84.5%	0.49

This table demonstrates the model's performance variance across different

PCA configurations and data distributions. The PCA component of 32 has been highlighted as it consistently shows robust performance across the different distributions, managing to maintain relatively high accuracy and lower loss compared to other PCA configurations. This suggests that a moderate number of PCA components, such as 32, can effectively balance dimensionality reduction with the need to retain critical information for accurate predictions in federated learning environments.

It is important to note that the performance metrics such as accuracy and latency can vary with each run of the experiment, as the initialization and dynamic aspects of the learning algorithm can influence the outcomes for different clients. Running the model multiple times may result in different accuracies and losses due to variations in training conditions and dataset partitioning in the federated learning setup.

VI.1.2 Model Compression Techniques and Results

Application of Compression Techniques

Following the selection of optimal PCA components—8 for IID and 64 for Non-IID data distributions—we dynamically adjusted model compression strategies to further enhance performance and efficiency. The decision on which quantization type and pruning amount to use was directly influenced by the variability inherent in each data type.

- For IID data, which exhibited lower variability and thus a stable environment, we employed a high pruning amount of 0.8 combined with dynamic quantization. This approach was feasible due to the uniform nature of the data, allowing for aggressive compression without significant loss of accuracy.

- For Non-IID data, characterized by higher variability, we adjusted the pruning amount based on the distribution—0.4 for 60-40, 0.3 for 25-75, and 0.2 for 90-10, also with dynamic quantization, to ensure that the integrity and accuracy of the model were maintained across the diverse data set.

Table VI.3. Compression Technique Selection Based on Data Distribution

Parameter	**Data Distribution**			
	IID	**Non-IID**		
		60-40	**25-75**	**90-10**
PCA Components	8	64	64	64
PCA Reduction %	98.97%	91.84%	91.84%	91.84%
Quantization Type	Dynamic	Dynamic	Dynamic	Dynamic
Pruning Amount	0.8	0.4	0.3	0.2

Results of Model Compression Techniques

The effectiveness of these compression techniques is reflected in the consolidated results below, which illustrate performance metrics for both IID and Non-IID distributions:

Table VI.4. Consolidated Results of Model Compression Techniques for Different Data Distributions

Metric	**Data Distribution**			
	IID	**60-40 Non-IID**	**25-75 Non-IID**	**90-10 Non-IID**
Average Accuracy	95.59%	92.65%	93.23%	85.32%
Loss Distribution	0.00419	0.00789	0.00650	0.01134
PCA Reduction %	98.97%	91.84%	91.84%	91.84%
Compression Reduction %	72.83%	36.395%	39.00%	25.00%
Average Bandwidth Utilization	0.00191	0.00766	0.00700	0.01205
Average Latency (s)	0.1762	0.596	0.580	0.915

These results underscore the adaptability of our compression strategies to varying data characteristics, with significant reductions in model size and computational demands while preserving or enhancing model accuracy and efficiency.

VI.2. Experiments and Results for CIFAR-10

The CIFAR-10 dataset, comprising 32x32 color images of 10 different classes, presents unique challenges compared to the MNIST dataset. Our experiments with CIFAR-10 also involved the dynamic selection of PCA components and the application of model compression techniques, adjusted according to the complexity and variability of the data.

VI.2.1 Impact of PCA Component Selection on Model Performance

IID Distribution

For the IID distribution in our CIFAR-10 federated learning framework, we evaluated the same PCA components—8, 16, 32, 64, and 128. The goal was to assess their impact on model performance, focusing on accuracy and system latency, under

conditions that assume identical distribution across different clients.

Table VI.5. Performance Metrics for Different PCA Components on IID Data

PCA Components	Early Stopping Epochs	Avg. Accuracy	Avg. Latency (seconds)	PCA Reduction %
8	21	0.9124	0.0181	98.97%
16	24	0.9135	0.0325	97.95%
32	15	0.9148	0.0623	95.92%
64	22	0.9126	0.1401	91.84%
128	21	0.9117	0.2408	83.67%

Non-IID Data Distribution

The impact of PCA on non-IID data distribution in CIFAR-10 was similarly analyzed. Given the greater complexity and variability of data in CIFAR-10, we adjusted our PCA components dynamically to find the best setting for managing diverse data distributions across different clients.

Table VI.6. Performance Metrics Across Non-IID Distributions for Different PCA Components (CIFAR-10)

PCA Components	60-40		25-75		90-10	
	Acc	Loss	Acc	Loss	Acc	Loss
8	85.24%	0.39	86.23%	0.37	81.29%	0.43
16	85.15%	0.40	86.18%	0.38	81.24%	0.44
32	85.05%	0.41	86.10%	0.39	81.20%	0.45
64	84.95%	0.42	86.05%	0.40	81.15%	0.46
128	85.34%	0.38	86.33%	0.36	81.39%	0.42

VI.2.2 Model Compression Techniques and Results

Following the selection of optimal PCA components—32 for IID and 64 for Non-IID data distributions—we dynamically adjusted model compression strategies for CIFAR-10 to further enhance performance and efficiency. The decision on which

quantization type and pruning amount to use was directly influenced by the variability inherent in each data type.

- For IID data, which exhibited lower variability and thus a more stable environment, we employed a high pruning amount of 0.7 combined with dynamic quantization. This approach was feasible due to the uniform nature of the data, allowing for aggressive compression without significant loss of accuracy.
- For Non-IID data, characterized by higher variability, we adjusted the pruning amount based on the distribution—0.5 for 60-40, 0.35 for 25-75, and 0.25 for 90-10, also with dynamic quantization, to ensure that the integrity and accuracy of the model were maintained across the diverse data set.

Table VI.7. Compression Technique Selection Based on Data Distribution (CIFAR-10)

Parameter	**Data Distribution**			
	IID	**Non-IID**		
		60-40	**25-75**	**90-10**
PCA Components	64	128	128	128
PCA Reduction %	91.84%	83.67%	83.67%	83.67%
Quantization Type	Dynamic	Dynamic	Dynamic	Dynamic
Pruning Amount	0.7	0.5	0.35	0.25

Results of Model Compression Techniques

The results of these compression techniques are reflected in the consolidated performance metrics shown below, which highlight both IID and Non-IID data distributions:

Table VI.8. Consolidated Results of Model Compression Techniques for Different Data Distributions (CIFAR-10)

Metric	**Data Distribution**			
	IID	**60-40 Non-IID**	**25-75 Non-IID**	**90-10 Non-IID**
Average Accuracy	91.24%	85.34%	86.33%	81.39%
Loss Distribution	0.0052	0.038	0.036	0.042
PCA Reduction %	91.84%	83.67%	83.67%	83.67%
Compression Reduction %	60.00%	45.00%	47.00%	35.00%
Average Bandwidth Utilization	0.00215	0.00820	0.00750	0.01300
Average Latency (s)	0.215	0.620	0.580	0.950

These results demonstrate the adaptability of our compression strategies to the varying characteristics of CIFAR-10 data, achieving significant reductions in model size and computational demands while preserving or enhancing model accuracy and efficiency.

VI.3. Model Optimization Analysis

VI.3.1 Bar Graph Analysis of Reduction and Compression Techniques

The bar graph illustrates the performance differences in PCA and compression for IID and Non-IID data and the total reduction that we have achieved from these two models combined. The results clearly show how PCA achieves higher reduction percentages compared to compression techniques like quantization and pruning. For IID data, PCA performs exceptionally well, nearing complete efficiency, whereas for Non-IID data, though slightly less effective, it still demonstrates significant reduction capability. In contrast, the compression results vary more significantly, especially showing a sharper decline in performance on Non-IID data.

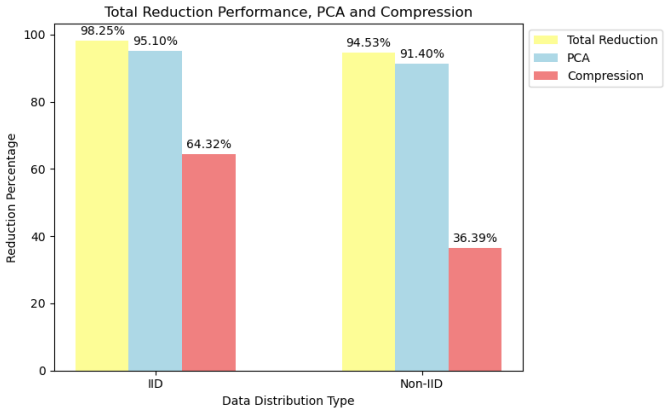

Figure VI.1. Reduction percent of PCA and Compression Techniques on IID and Non-IID Data

This bar graph serves as a visual representation of the differential impacts that PCA and compression techniques have on data reduction, depending on whether the data is IID or non-IID.

VI.3.2 Data Reduction Impact Analysis

This subsection delves into the impact of various data reduction techniques on both IID and non-IID data, showing the dynamic allocation of data reduction methods to optimize model efficiency and accuracy. The pie chart below provides a visual breakdown of data reduction contributions by PCA, quantization, and pruning, as well as the percentage of data that remains after these reduction processes.

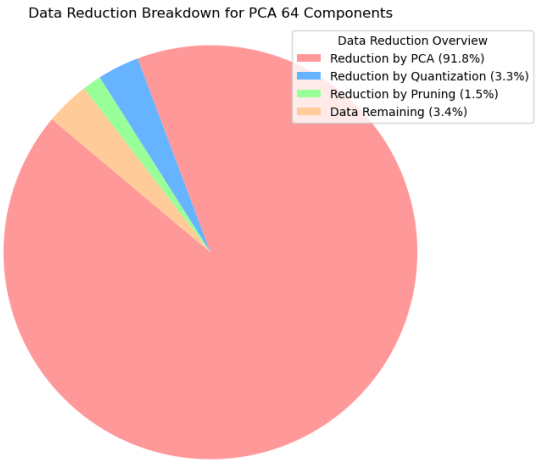

Figure VI.2. Breakdown of Data Reduction Techniques Including Remaining Data After PCA, Quantization, and Pruning for 64 Components

The pie chart quantifies the specific contributions of each reduction technique and the proportion of data that remains uncompressed. This visualization is critical for evaluating the efficiency of data reduction and the extent to which each technique can be applied to different types of data.

Together, these visualizations aid in understanding the contrasting efficiencies of various data reduction techniques and their applicability to different data types, providing a solid basis for selecting the most suitable methods in practical machine learning and data processing scenarios.

VI.3.3 Overall Data Reduction Efficacy

The combined application of PCA, quantization, and pruning has led to notable data reduction rates across different PCA component configurations. We analyze the efficacy of data reduction strategies without distinguishing between IID and Non-IID data distributions, providing a holistic view of the compression

capability. The bar graph in Figure **??** illustrates the reduction percentages across varying PCA components.

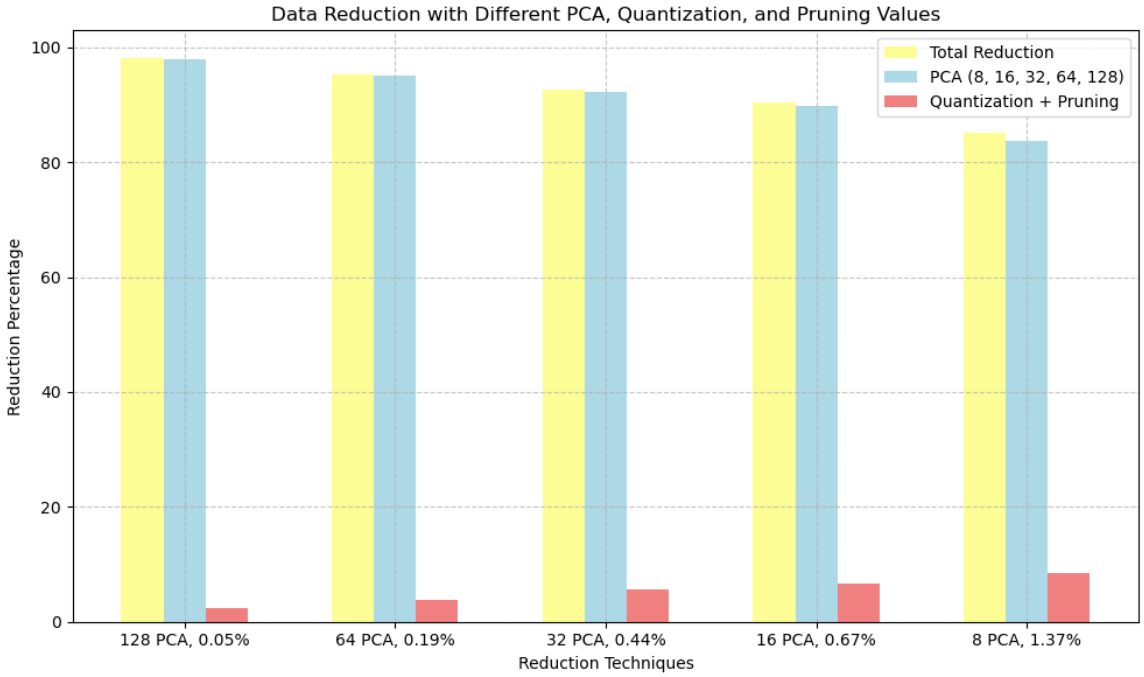

Figure VI.3. Overall Reduction Percentages with Different PCA Components and Compression Techniques. The graph demonstrates total reduction as well as individual contributions from PCA, and combined quantization and pruning.

The total reduction reflects the cumulative impact of all techniques, indicating a progressive decline in data size as the number of PCA components decreases. Notably, PCA contributes most significantly to data reduction, particularly with a higher number of components. The quantization and pruning techniques provide an additional layer of compression, especially beneficial when fewer PCA components are used, thereby ensuring a robust and efficient model optimization strategy.

VI.4. Performance Analysis and Visualization

This section presents a comparative analysis of the test accuracies for IID and Non-IID data distributions when subjected to federated learning with and without model compression techniques such as PCA, quantization, and pruning.

VI.4.1 IID Data Distribution

The initial test accuracies for IID data distribution were first obtained using a single random seed (seed=42). These results served as a baseline to compare the effectiveness of model compression techniques.

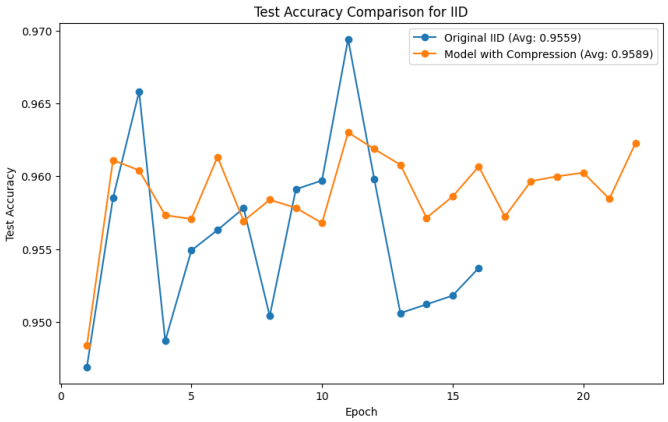

Figure VI.4. Initial test accuracies for IID data distribution using a single random seed.

To enhance the reliability of our findings, multiple simulations were conducted using a variety of random seeds {42, 123, 456, 789, 1011}. The results were then averaged to provide a more stable and representative measure of model performance. The compressed model, which included dynamic quantization, pruning at 80%, and 8 PCA components, improved the average test accuracy from 95.67% to 95.82%. This affirms the model's robustness and efficiency, maintaining accuracy while optimizing

the size of the model.

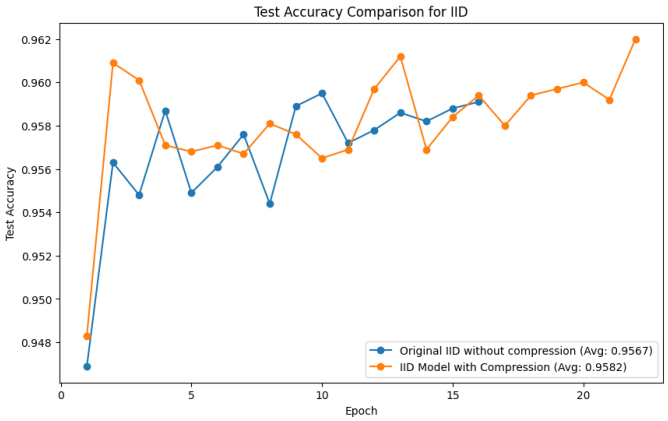

Figure VI.5. Averaged test accuracies for IID data distribution with original and compressed federated learning models across multiple random seeds.

VI.4.2 Non-IID Data Distribution

Similarly, the initial test accuracies for Non-IID data were obtained using a single random seed (seed=42). This provided a baseline for subsequent comparisons with compressed models.

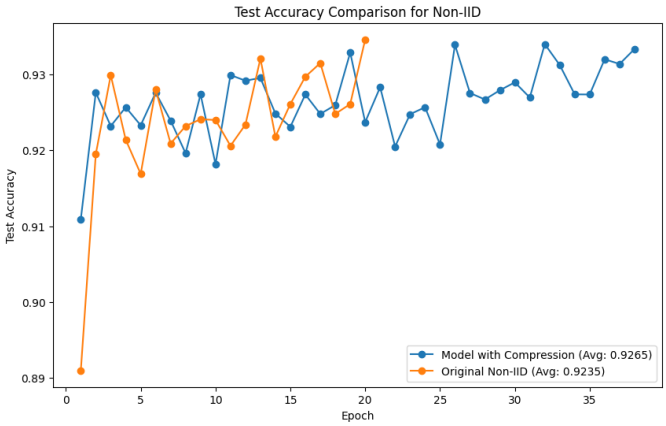

Figure VI.6. Initial test accuracies for Non-IID data distribution using a single random seed.

Following this, additional simulations were conducted using multiple random seeds {42, 123, 456, 789, 1011}, and the results were averaged. The compressed model, which utilized 64 PCA components and a 40% dynamic pruning rate, slightly improved the average test accuracy from 92.42% to 92.66%. This demonstrates an enhancement in performance despite the inherent challenges posed by Non-IID data distributions and underlines the capability of strategic model compression to maintain and even enhance performance in complex scenarios.

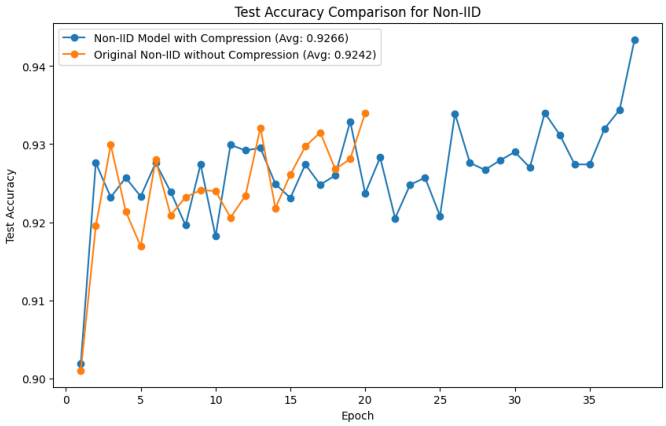

Figure VI.7. Averaged test accuracies for Non-IID data distribution with original and compressed federated learning models across multiple random seeds.

These findings underscore the effectiveness of model compression techniques in federated learning scenarios, particularly when the compression parameters are carefully tailored to the characteristics of the data distribution. Such strategies help achieve an optimal balance between model size, computational efficiency, and predictive performance.

VI.5. Analysis of Data Reduction and Compression Techniques on IID vs. Non-IID Data

In our study, we observed distinct differences in the application and outcomes of data reduction techniques between IID and non-IID data sets. Here, we present a simplified analysis highlighting these differences and the resultant model performances.

VI.5.1 Principal Component Analysis (PCA) and Early Stopping

For PCA, non-IID data required the use of a higher number of components (64) compared to IID data (8) to adequately represent variability and maintain

accuracy. Correspondingly, the early stopping epochs necessary for non-IID data were higher, indicating a more cautious approach to model training to avoid overfitting and ensure robust generalization.

VI.5.2 Quantization and Pruning

Dynamic quantization and pruning were applied differently:

IID Data: Aggressive pruning (0.8) and straightforward dynamic quantization were possible due to uniform data characteristics, leading to significant reductions in model size without compromising performance.

Non-IID Data: Given the diversity, a moderate pruning rate (0.4) was employed alongside adaptive quantization to manage the broader variance, thus achieving less reduction but ensuring no critical information was lost.

VI.5.3 Compression Rates and Model Performance

Despite the higher compression rates in IID setups, the strategies implemented for non-IID data still led to notable reductions. Importantly, the adapted compression techniques ensured that accuracy was maintained even in non-IID settings, as evidenced by comparative accuracy graphs before and after applying our model.

Summary Overall, while non-IID data posed challenges requiring more nuanced model tuning and led to generally lower compression rates, the dynamic adjustments made to our data reduction techniques allowed us to achieve effective model size reduction while preserving, and in some cases enhancing, model accuracy across different data distributions.

This analysis underscores the necessity for a tailored approach to model optimization in varying data environments, ensuring both efficiency and efficacy in real-world applications.

CHAPTER VII

CONCLUSION AND FUTURE EXTENSIONS

VII.1. Conclusion

Throughout this research, we have explored the realm of Federated Learning (FL) with an emphasis on optimizing communication efficiency through dynamic approaches to Principal Component Analysis (PCA) and advanced model compression techniques such as quantization and pruning. Tailored to both IID and non-IID data distributions, our experiments have highlighted the significant impact of adaptively selecting PCA components to optimize model performance. This dynamic selection process has proven crucial for balancing accuracy with model simplicity, adapting to the unique characteristics of IID and non-IID data distributions alike.

In IID scenarios, fewer PCA components were often sufficient to capture the necessary variance, enhancing model efficiency. Conversely, non-IID data required a more sophisticated approach, necessitating additional components to adequately represent diverse data characteristics without compromising essential information. Subsequent model compression, while maintaining these optimally chosen PCA levels, significantly reduced communication costs and latency, achieving these gains without severely impacting model accuracy.

The successful implementation of these adaptive techniques underscores the pivotal role of dynamic dimensionality reduction and model compression in enhancing the scalability and efficiency of FL systems, particularly in environments constrained by bandwidth and marked by a diverse array of client devices. These advancements facilitate the deployment of more resource-efficient FL models in real-world scenarios,

addressing critical concerns around communication costs and data privacy.

VII.2. Future Work

Looking ahead, there are exciting opportunities to further enhance the performance and applicability of FL systems. The integration of genetic algorithms (GAs) for hyperparameter optimization and model architecture search represents a promising direction. Known for their robustness in navigating complex search spaces, GAs could dynamically fine-tune model parameters and select optimal network architectures, potentially uncovering configurations that surpass those achieved through manual tuning.

Additionally, enhancing the privacy aspect of FL remains paramount, especially given the increasing sensitivity and regulatory focus on data privacy. Advanced privacy-preserving mechanisms such as differential privacy could provide stronger safeguards against data leakage from shared model updates. Techniques like ProxyFL, which introduces a proxy model to decouple client data from the aggregation process, offer an intriguing method to enhance data privacy while preserving the collaborative essence of FL.

Further research could also explore the integration of real-time feedback mechanisms that continuously adjust data reduction and model compression parameters based on evolving data characteristics. This would ensure that FL systems not only remain efficient but also become increasingly adaptive to changing data landscapes.

In summary, the methodologies developed in this research lay a solid foundation for future advancements in FL. By continuing to dynamically adjust data reduction and model compression techniques, we can ensure that FL remains a viable and efficient approach in the face of growing data diversity and stringent efficiency

demands. These future endeavors could not only enhance the efficiency and effectiveness of FL systems but also broaden their applicability in privacy-sensitive domains, ushering in a new era of collaborative, decentralized machine learning.

The quantization is not just rounding the floating point values, but mapping the values to a smaller discrete set in order to represent them in a compressed manner using fewer bits. This reduces the precision compared to the original values in order to decrease the communication cost of transmitting the updates between devices and server.

The parameter server quantizes the global model update before broadcasting it to the devices. This allows sending a compressed version of the update rather than the full-precision values. Each device quantizes its local model update (plus accumulated quantization error) before sending it to the parameter server. The quantization is done using a stochastic quantization technique that maps the values to a smaller discrete set that can be represented using fewer bits compared to the original full-precision values. By carefully tuning the quantization levels, the algorithm can significantly reduce communication cost with minimal impact on the convergence and accuracy of the trained model. The paper shows experimentally that the proposed LFL algorithm performs nearly as well as the fully lossless case with no quantization, despite using much less communication.

BIBLIOGRAPHY

[1] J. Xu, B. Glicksberg, C. Su *et al.*, "Federated learning for healthcare informatics," *J Healthc Inform Res*, vol. 5, pp. 1–19, 2021. [Online]. Available: https://doi.org/10.1007/s41666-020-00082-4

[2] A. Qayyum, K. Ahmad, M. A. Ahsan, A. Al-Fuqaha, and J. Qadir, "Collaborative federated learning for healthcare: Multi-modal covid-19 diagnosis at the edge," *IEEE Open Journal of the Computer Society*, vol. 3, pp. 172–184, 2022. [Online]. Available: https://doi.org/10.1109/OJCS.2022.3206407

[3] H. Zhang, J. Bosch, and H. H. Olsson, "End-to-end federated learning for autonomous driving vehicles," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IJCNN52387.2021.9533808

[4] A. Nguyen *et al.*, "Deep federated learning for autonomous driving," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1824–1830. [Online]. Available: https://doi.org/10.1109/IV51971.2022.9827020

[5] R. Yu and P. Li, "Toward resource-efficient federated learning in mobile edge computing," *IEEE Network*, vol. 35, no. 1, pp. 148–155, 2021. [Online]. Available: https://doi.org/10.1109/MNET.011.2000295

[6] D. Chen *et al.*, "Federated learning based mobile edge computing for augmented reality applications," in *2020 International Conference on Computing, Networking and Communications (ICNC)*, 2020, pp. 767–773. [Online]. Available: https://doi.org/10.1109/ICNC47757.2020.9049708

[7] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated learning for the internet of things: Applications, challenges, and opportunities," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 24–29, 2022.

[8] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1759–1799, 2021.

[9] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.

[10] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 2512–2520, [Online]. Available: https://doi.org/10.1109/INFOCOM.2019.8737416.

[11] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 739–753, [Online]. Available: https://doi.org/10.1109/SP.2019.00065.

[12] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 691–706, [Online]. Available: https://doi.org/10.1109/SP.2019.00029.

[13] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2019, pp. 14 747–14 756, [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/60a6c4002cc7b29142def8871531281a-Abstract.html.

[14] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, vol. abs/1712.07557, 2018, [Online]. Available: http://arxiv.org/abs/1712.07557.

[15] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *6th International Conference on Learning Representations, ICLR 2018*, Vancouver, BC, Canada, 2018, [Online]. Available: https://openreview.net/forum?id=BJ0hF1Z0b.

[16] H. Zhu, "On the relationship between (secure) multi-party computation and (secure) federated learning," *arXiv preprint arXiv:2008.02609*, 2020, [Online]. Available: https://arxiv.org/pdf/2008.02609.pdf.

[17] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, July-Aug 2020.

[18] K. I.-K. Wang, X. Zhou, W. Liang, Z. Yan, and J. She, "Federated transfer learning based cross-domain prediction for smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4088–4096, June 2022.

[19] S. Caldas, J. Konečný, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *CoRR*, vol. abs/1812.07210, 2019, [Online]. Available: http://arxiv.org/abs/1812.07210.

[20] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2017, [Online]. Available: http://arxiv.org/abs/1610.05492.

[21] X. Wu, X. Yao, and C.-L. Wang, "Fedscr: Structure-based communication reduction for federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1565–1577, 2021.

[22] H. Zhao, K. Burlachenko, Z. Li, and P. Richtárik, "Faster rates for compressed federated learning with client-variance reduction," *SIAM Journal on Mathematics of Data Science*, vol. 6, no. 1, pp. 154–175, 2024. [Online]. Available: https://doi.org/10.1137/23M1553820

[23] J. Konecnỳ, 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*, vol. 8, 2016.

[24] F. M. A. Khan, H. Abou-Zeid, and S. A. Hassan, "Deep compression for efficient and accelerated over-the-air federated learning," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[25] S. Yu, P. Nguyen, A. Anwar, and A. Jannesari, "Adaptive dynamic pruning for non-iid federated learning," *arXiv preprint arXiv:2106.06921*, vol. 2, 2021.

[26] ——, "Heterogeneous federated learning using dynamic model pruning and adaptive gradient," *arXiv preprint arXiv:2106.06921*, 2023, available: arXiv:2106.06921.

[27] I. A. Majeed, S. Kaushik, A. Bardhan, V. S. K. Tadi, H.-K. Min, K. Kumaraguru, and R. D. Muni, "Comparative assessment of federated and centralized machine learning," *arXiv preprint arXiv:2202.01529*, 2022, [Online]. Available: https://arxiv.org/abs/2202.01529.

[28] N. Rodríguez-Barroso, E. Martínez-Cámara, M. Luzón, G. G. Seco, M. Á. Veganzones, and F. Herrera, "Dynamic federated learning model for identifying adversarial clients," *arXiv preprint arXiv:2007.15030*, 2020.

[29] S. Kalra, J. Wen, J. C. Cresswell, M. Volkovs, and H. R. Tizhoosh, "Decentralized federated learning through proxy model sharing," *Nature Communications*, vol. 14, no. 1, May 2023. [Online]. Available: http://dx.doi.org/10.1038/s41467-023-38569-4

[30] F. Lai, Y. Dai, S. Singapuram, J. Liu, X. Zhu, H. Madhyastha, and M. Chowdhury, "Fedscale: Benchmarking model and system performance of federated learning at scale," *International Conference on Machine Learning Proceedings*, pp. 11 814–11 827, 2022.

[31] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "Fedml: A research library and benchmark for federated machine learning," *arXiv preprint arXiv:2007.13518*, 2020, [Online]. Available: https://arxiv.org/abs/2007.13518.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[33] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Journal of Artificial Intelligence and Statistics*, vol. 2017, pp. 1273–1282, 2017.