Privacy Preserving Machine Learning

A Synopsis Report submitted
in the fulfillment of the requirement for The degree of

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

Ashutosh Ghanto (18BCS014)

Laksh Gangwani (18BCS046)

Rupesh Kumar (18BCS081)

Avinash Kumar (18BCS015)

Sachin Verma (18BCS085)

Under the guidance of

Dr. Malay Kumar

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

DHARWAD

CERTIFICATE

It is to certify that the work contained in the project report titled "**Privacy Preserving Machine Learning**" by Asutosh Ghanto (18BCS014), Avinash Kumar(18BCS015), Laksh Gangwani (18BCS046), Rupesh Kumar (18BCS081) and Sachin Verma (18BCS085) has been submitted for the fulfillment of the requirement for the degree of "Bachelor of Technology in Computer Science & Engineering". Their work has been found satisfactory and hereby approved for the submission.

Signature of the Supervisor
Dr. Malay Kumar
Assistant Professor
Department of Computer Science and Engineering
IIIT-Dharwad

Declaration by candidate

We hereby certify that the work embodied in this report entitled " **Privacy Preserving Machine Learning ”** by us in the partial fulfillment of Major project for Eighth Semester B.Tech submitted to Indian Institute of Information Technology, Dharwad is an authentic record of our own work carried out under supervision of **Dr. Malay Kumar**. The content presented in this discussion has not been submitted by us in any other university/institute for the award of any other degree or diploma. Responsibility of any plagiarism related issue stands solely with us.

Acknowledgement

With a deep sense of gratitude, we express sincere thanks to our supervisor, **Dr. Malay Kumar**, Assistant Professor, Computer Science and Engineering, Indian Institute of Information Technology Dharwad, for his valuable guidance, advice and help with different views of application, development and scalability. His guidance has been a cornerstone in Deep Learning Techniques. We are able to accomplish this project with his valuable directions and suggestions.

Sincerely,

ASHUTOSH GHANTO (18BCS014)

LAKSH GANGWANI (18BCS046)

RUPESH KUMAR (18BCS081)

SACHIN VERMA (18BCS085)

AVINASH KUMAR (18BCS015)

Approval Sheet

This project report entitled ("**Privacy Preserving Machine Learning"**) by (Rupesh Kumar), (Asutosh Ghanto), (Avinash Kumar), (Laksh Gangwani) and (Sachin Verma) is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

Signature of the Supervisor
Dr. Malay Kumar
Assistant Professor
Department of Computer Science and Engineering
IIIT-Dharwad

Head of Department
Dr. Uma Sheshadri
Assistant Professor
Department of Computer Science and Engineering
IIIT-Dharwad

Date : ……
Place : ……..

Abstract

Distributed learning has emerged as a useful tool for analyzing data stored in multiple geographic locations, particularly when the distributed data sets are large and difficult to move around, or when the data owner is hesitant to put data into the Cloud because of privacy concerns. Only the locally computed models are uploaded to the fusion server in distributed learning, but this may still cause privacy concerns because the fusion server could use its observations to perform various inference attacks. To address this issue, we propose a secure distributed learning system that aims to prevent direct exposure of computed models to the fusion server by utilizing the additive property of partial homomorphic encryption.

In this project, we want to demonstrate a secure way of sharing medical data from a large number of hospitals with a company X that helps these hospitals use machine learning techniques without ever having to reveal their actual data. We solve the problem of determining whether breast cancer is benign or malignant in our project. For our use case, we use the scikit-learn dataset load-breast-cancer. We build a Logistic Regression model because the task is binary classification and use Homomorphic Encryption for shared training and encryption.

TABLE OF CONTENTS

Chapter 1

1.1 Introduction 1

Chapter 2 Literature Review

2.1 Overview 3

2.2 Research Methodologies 4

Chapter 3 Homomorphic Encryption

3.1 Types of Homomorphic Encryption 19

3.2 Deep Dive 20

Chapter 4

4.1 Methodology and Implementation 22

Chapter 5 Results

5.1 Breast Cancer dataset 24

5.2 Grad Admission dataset 25

5.3 Acute Inflammation dataset 26

5.4 Fed-Avg and Fed-Prox with MNIST dataset 28

Chapter 6 Conclusion

6.1 Conclusion 44

References 45

CHAPTER 1

INTRODUCTION

Machine learning is an important technique in computer science to solve problems that normal classical algorithms fail to solve. It tries to use data to gather inferences rather than using explicit code. But It isn't always viable to get a huge amount of data to solve these problems due to the sensitive nature of some special kinds of data like financial data,health data,private messages etc.

So new computation techniques were invented to solve this kind of problems with data security, data privacy of the user and to also abide by the security compliance by the governments to use data. In this we would be discussing various techniques to ensure security and privacy of data generated by the user while training a machine learning model.

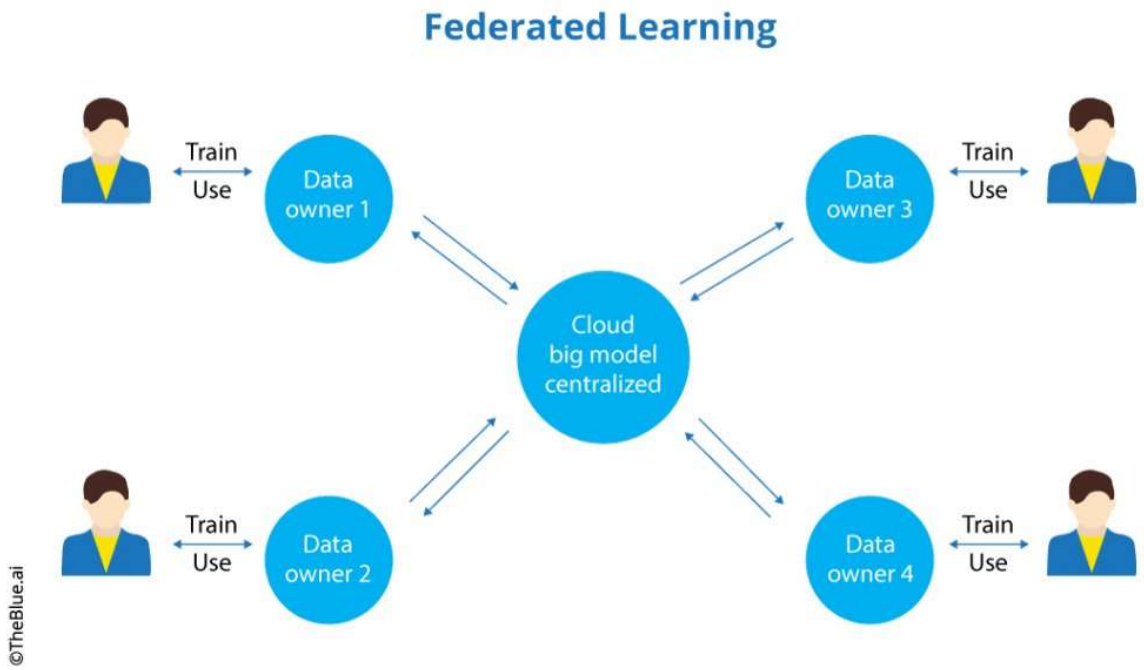


Fig 1: Federated Learning

Massive amounts of data are now dispersed across several geographical locations. The most common approach for mining data for useful analytics is to gather all of the data in a centralized location and train an AI model on the aggregated data. However, this approach may not be feasible in a variety of situations. Sharing data across national or organizational boundaries, in particular, may not always be permitted.

Significant work in this area predates the coining of the term "federated learning." Many research communities (including cryptography, databases, and machine learning) have long sought to analyze and learn from data that is distributed among many owners without exposing

it. Beginning in the early 1980s, Agrawal and Srikant and Vaidya et al. are early examples of work that attempted to learn from local data utilizing a centralized server while retaining anonymity. On the other hand, we are aware of no single effort that explicitly addresses the entire range of FL issues, even though the phrase federated learning has been around for a long time. was coined. As a result, the term federated learning serves as a handy shorthand for a set of characteristics, constraints, and challenges that frequently co-occur in applied machine learning problems on decentralized data where privacy is a top priority.

Because federated learning is typically used to train models across multiple organizations, the basic approach is to share model parameters rather than training data. It can integrate artificial intelligence insights from data found in various systems and locations by migrating models rather than data for training. While existing federated learning approaches tend to focus on performance metrics like the time it takes to train a model optimally over bandwidth-constrained networks or achieving model accuracy metrics comparable to centralized learning, trust is an important aspect of federated learning across multiple organizations. The current work assumes that all sites have complete trust in one another and that each site is comfortable sharing model parameters with the others. The organizations involved, on the other hand, may not have complete trust in one another.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

In The Present Time, The Rapid Increase in Development of Mobile devices, and Vehicles Connected With Smart Devices (ex : Mobile), Computing edge devices and sensors for data Collection has all brought the explosive growth in data, That highly power up the Traditional Methods of Machine Learning. However, these common methods generally requires the centralized model training by storing the data at a single machine or the central cloud data centers, that leads to certain problems such as computational efficiency, Data Privacy etc. Due to the development of storage and Computing capabilities of distributed computing edge devices, using increased computation power on the edge devices became an applicable solution.

In the Federated Learning system, the local model training is used and the data created by the edge devices need not to be shared. Instead, the weight updates are sent to the central aggregation server for the purpose of generating the global model. This system solves the problem that the model in the traditional Machine Learning can be trained and delivered only on the single central server. The Federated Learning Theory has been explored, After the concept was first applied and introduced by Google in 2017, there has been numerous Federated Learning frameworks, architectures and solutions proposed to solve the real world problems.

First, we will provide the literature review within the Field of Federated Learning systems.
We then identify and categorize the existing literature into different application-domains according to the problems Formulated and solved. Based upon limitations and challenges identified in our literature review, we will deal with the literature review with the help of research questions.

2.2 Research Methodologies

Our purpose of review is to present the overview of contemporary research over the empirical results and the solutions regarding the Federated Learning Those had been reported in existing literatures.

We will address Some research questions as well.

We have followed the protocols as follows :

Background → Requirement Analysis → Architecture Design → Implementation & Evaluation

2.2.1 Background Understanding

Fig 2. What is FL?

To Answer question 1, each study's definition of federated learning we recorded.

This inquiry will assists in understanding

(1) what FL is and (2) what scholars have to say about it

The frequency of words that appear in each study's original definition of federated learning is depicted in Fig 2 as a word cloud. The terms "distribute," "local," "device," "share," "client," "update," "privacy," "aggregate," and "edge" were commonly used. To help us better answer Q1, we've divided federated learning definitions into five categories:

(1) training settings, (2) data distribution, (3) orchestration, (4) client kinds, and (5) data partitioning.

Firstly, most academics defined federated learning as creating a decentralized or distributed learning process across several clients in a training scenario. This may be seen in the fact that "training a model on numerous clients" was the most often referenced keyword in Q1. Other terms used to characterize training environments include "distributed," "collaborative," and "many parties/clients." "Only communicating model modifications to the central server" and "building a global model on the central server" are the other two characteristics that characterize how a federated learning system achieves distributed model training. This also shows how academics differentiate between federated learning and traditional and distributed machine learning.

Secondly, data distributions can be used to explain federated learning. "Data generated locally" and "data stored decentralized" are the keywords that were mentioned in the studies. Client devices in various geographical areas acquire and store data. As a result, it has data attributes that are non-IID and imbalanced. In order to ensure data privacy, the data is decentralized and not shared with other clients. We'll go over client data distribution in more detail in a later section.

Thirdly, researchers look into federated learning from the perspective of training process orchestration. A central server orchestrates the training procedures in traditional federated learning. Initializing a global model, distributing global models to participating client devices, collecting trained local models, and aggregating the gathered local models to update the global model are the tasks. Researchers intuitively view the use of a single central server to be a potential single-point-of-failure. As a result, decentralized mechanisms for exchanging model updates are investigated, and decentralized data governance is developed.

Fourthly, we perceive cross-device and cross-silo federated learning in terms of client kinds. Cross-device federated learning uses a large number of smart devices to form a large-scale dispersed network that may be used to cooperatively train a model for the same applications. Two examples of applications are word suggestions on mobile devices and human behavior recognition. The options have been expanded to include cross-silo applications that limit data sharing across firms. Data security regulations, for example, prevent a hospital's information from being shared with other hospitals. Cross-silo federated learning conducts local model training using data from each hospital in order to enable machine learning in this setting.

Finally, we discovered three types of data partitioning: horizontal, vertical, and federated transfer learning. Horizontal federated learning, also known as sample-based federated learning, is used when the datasets have the same feature space but separate sample ID spaces. When two or more datasets share the same sample ID space but not the same feature space, vertical federated learning, also known as feature-based federated learning, is utilized. Data partitioning is taken into account in federated transfer learning, where two datasets only partially overlap in sample space or feature space. Its objective is to develop models that can be applied to both datasets.

Purpose is to learn more about the advantages of federated learning. The responses are categorized based on the non-functional requirements of federated learning adoption.
Data privacy and communication efficiency are the two main motivations for federated learning adoption. Data privacy is preserved with federated learning since no raw local data leaves the device.

Furthermore, by transmitting only model parameters or gradients, federated learning improves communication efficiency. Scalability is aided by high data privacy and transmission efficiency. As a result, more clients are encouraged to participate in the training process. The data distribution with data volume and class distribution variance among devices is known as statistical heterogeneity (i.e. Non-IID). In essence, the data is dispersed widely across client devices, each of which only stores a limited amount of data and has unbalanced data classes.
that aren't reflective of the data distribution as a whole Local models that are trained independently on these devices tend to be over-fitted to their local data.

As a result, federated learning is used in a group environment to train local models into a global model. Devices with varied resources are said to have system heterogeneity (e.g, computation, communication, storage, and energy). Federated learning can help solve this problem by allowing local model training and just communicating model changes, which saves bandwidth and energy.

Another reason is the high computing efficiency. With a large number of participating clients and rising compute capabilities, federated learning can achieve good model performance and computation efficiency. Data storage efficiency is ensured by independent on-client training using locally generated data.

Top 3 Phases : "Model training" is the most frequently reported machine learning process, followed by "data collecting" and "data cleaning." The methodologies in traditional machine learning systems are essentially identical to these three stages of federated learning. The distributed model training tasks, decentralized data storage, and non-IID data dissemination are the main differences. Only Google acknowledged model inference and deployment, and only for Google keyboard applications.

TensorFlow Lite's on-device inference is described as constructing and deploying an on-device inference model utilizing a model checkpoint from the server. It uses the same featurization approach that was originally used to log training samples on-device. Prior research, on the other hand, hasn't looked into deployed model monitoring (for example, dealing with performance degradation) or project management (e.g., model versioning). Because most studies focused on data processing and model training optimization, we deduce that federated learning research is still in its early stages.

Phases: The model training step is the one that gets the most attention. Only Google contains discussions about model deployment (e.g., deployment methodologies) and model inference, with data pre-processing, feature engineering, and model evaluation mentioned in a few studies. Model monitoring (e.g., dealing with performance degradation) and project management are not covered in the current research (e.g., model versioning). To build federated learning systems at the production level, more research is required.

2.2.2 Requirement Analysis

The requirements of federated learning systems are examined after the background understanding stage. Because functional needs are application-specific, we concentrate on non-functional requirements.

Despite the fact that sending model updates rather than raw data can save money on communication, federated learning systems still have to go through numerous update iterations to obtain convergence. As a result, methods for reducing communication rounds are being investigated. In addition, cross-device federated learning must be able to support a high number of client devices. As a result, due to bandwidth constraints, some customers may drop out. These dropouts potentially have two detrimental consequences for federated learning systems: decrease the amount of data available for training; lengthen the training period. The dropout

problem can be solved by discarding the global model, which is composed of a small number of local models.

By aggregating local models trained locally on client devices, federated learning is effective in dealing with statistical and system heterogeneity concerns. However, the method is still in its early stages, with issues such as how to handle non-IID while maintaining model performance remaining unsolved. People are also concerned about the security of their data and client devices. In federated learning, data security refers to a system's ability to ensure that only authorized parties have access to data. Despite the fact that federated learning systems prevent raw data from leaving local devices, back-tracing gradients can be used to recover private data. The majority of papers in this category are concerned with information leakage caused by changes in local model gradients.

The degree of security against dishonest and malevolent client devices is referred to as client device security. By disrupting the training process or sending false updates to the central server, rogue devices in the training process could taint the overall model performance. Furthermore, client device misbehavior, whether deliberate or inadvertent, may degrade system reliability.
Because model performance is heavily reliant on the number of participating clients, client motivatability is presented as an element to investigate. More clients engaging in the model training process means more data and compute resources are supplied to the process.
The majority of system dependability issues are caused by adversarial or byzantine attacks on the central server, which expose the single point of failure. Some research focuses on federated learning systems' system performance, which includes issues like processing efficiency, energy efficiency, and storage efficiency. The efficient resource management of federated learning systems is much more important in resource-constrained circumstances.

Auditing tools are used to track client device behavior, local model performance, and system runtime performance to increase system auditability. Finally, the model performance constraint is studied, and scalability is highlighted as a difficulty in federated learning. The model performance of federated learning systems is heavily influenced by the number of participants and the data volume of each participating client in the training process. Furthermore, model performance is constrained due to the non-IID data.

Data and applications: Federated learning is frequently utilized in systems that deal with image, structured, and text data. Graphs and sequential data are rarely used due to their data

characteristics. Only a few production-level apps are also accessible. The great majority of applications are still simulations or proof-of-concept models. This issue contributes to the requirement analysis phase of the software development lifecycle, where we identify the various applications and data types that have used federated learning.

The federated learning model's performance is influenced by the dispersion of client data. When aggregating models, the distribution of the dataset on each client should be considered. Numerous investigations have been conducted into the FedAvg technique extensions for model aggregation, with a focus on Non-IID difficulties. This topic adds to the requirement analysis phase, in which we determine the properties of various data distribution strategies that influence the federated learning model's performance.

2.2.3 Architecture Design

Model aggregation approaches address communication efficiency, statistical heterogeneity, system heterogeneity, client device security, data security, system performance, scalability, model performance, system auditability, and system dependability.

Researchers have proposed aggregation methodologies such as selective aggregation, aggregation scheduling, adaptive aggregation, temporally weighted aggregation, controlled averaging algorithms, iterative round decline, and shuffling model aggregation. These methods are designed to improve efficiency and scalability by lowering communication costs and latency; managing device computation and energy resources to address system heterogeneity and performance issues; and selecting high-quality models for aggregation based on model performance. Several academics have proposed secure aggregation as a solution to data security and client device security issues.

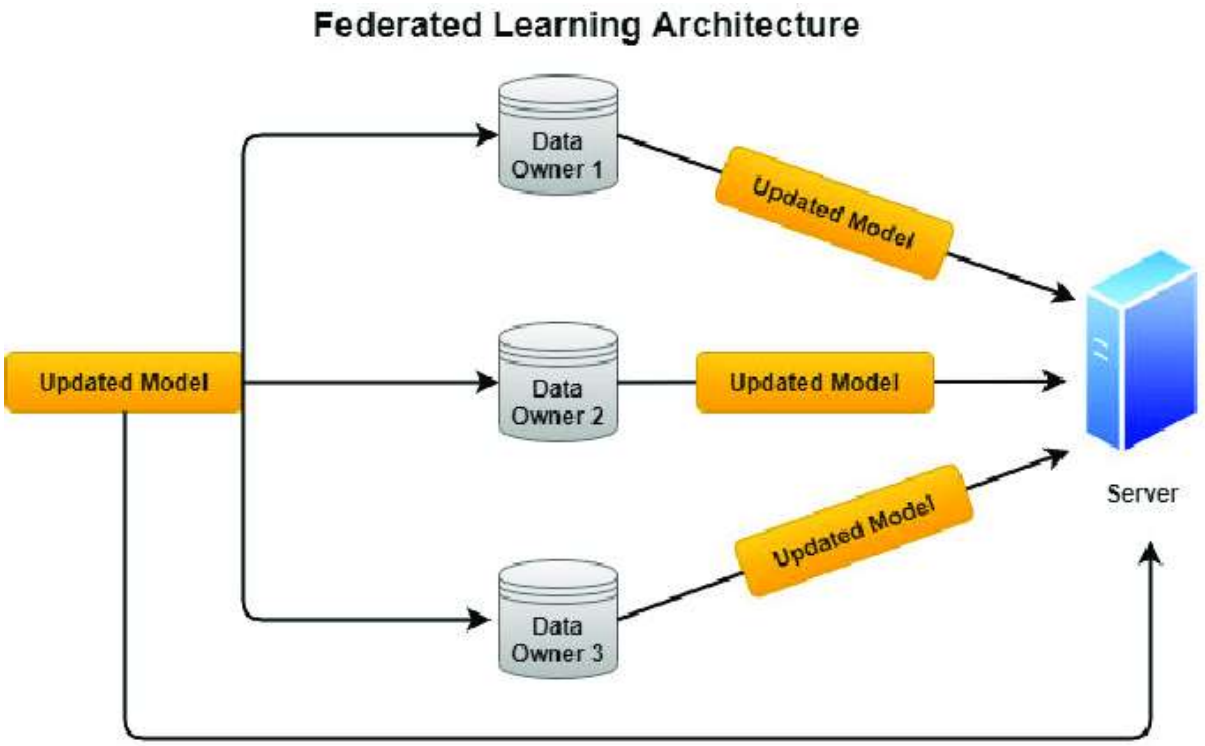


Fig 3: Architectural Design

To address system reliability issues, decentralized aggregation is a type of model aggregation in which the central server is removed from federated learning systems. A peer-to-peer system, one-hop neighbors collective learning, and Online Push-Sum (OPS) techniques can all be used to achieve decentralized aggregation.

According to studies, the second most popular strategy is training management. Statistical heterogeneity, system heterogeneity, communication efficiency, customer motivation, and client device security are just a few of the applications. To deal with statistical heterogeneity, researchers have proposed a variety of training methodologies. Clustering of training data, multistage training and finetuning of models, brokered learning, distributed multitask learning, and edge computing are just a few of the techniques used. Increasing the amount of training data while reducing skewness and maintaining data dispersion are the objectives.

The incentive system, Another strategy that is frequently recommended is, which is a tool for increasing client motivation. If there are no benefits to donating their data and resources, owners of data and devices are not required to participate in the training process. To persuade data and device owners to participate in model training, incentive schemes may be used. The amount of computation, communication, and energy resources available, the performance of the local model, the quality of the data provided, the honest behavior of client devices, and the

dropout rate of a client node can all influence incentives. A central server or a blockchain could be used to host the proposed incentive systems.

To address system heterogeneity and communication efficiency, resource management is used to oversee and optimize computation resources, bandwidth, and energy consumption of participating devices in a training process. The proposed techniques use control algorithms, reinforcement learning, and edge computing technologies to maximize resource consumption and improve system efficiency. Model compression procedures are used to reduce data size and communication costs that occur during model changes in order to improve communication efficiency. Model compression can help with scalability because it can be used in bandwidth-constrained and latency-sensitive scenarios.

Privacy preservation mechanisms are introduced to protect data security: prevents information (model parameters or gradients) from being leaked to unauthorized parties, and client device security: protects the system from being infiltrated by dishonest nodes. Differential privacy, which involves adding gaussian noise to features before changing the model, is a well-known data security strategy. To keep client devices secure, secure multiparty computations like homomorphic encryption are used in conjunction with a local differential privacy approach. Client data privacy is protected by introducing noise to model parameter data received by each client, whereas homomorphic encryption simply allows the central server to homomorphically generate the global model based on the encrypted local updates.

Encrypting model parameters or gradients prior to sharing models between the server and client nodes is also offered as a security protection solution to address difficulties with both client device security and data security. To improve communication efficiency and system performance, **communication coordination methods** are presented. Using communication tactics such as multi-channel random access communication or over-the-air computation approaches, wireless federated training systems can achieve faster convergence. To address model performance concerns, statistical heterogeneity issues, system auditability constraints, and communication efficiency limitations, feature fusion or selection procedures are applied. By simply aggregating models with the selected characteristics, feature selection approaches are utilizedFederated learning has piqued the curiosity of academics and industry alike. The results show that the majority of the recognised motivations for federated learning appear to be the most researched research topics in the field. Model aggregation, training management, incentive mechanisms, privacy preservation, and resource management are the top five

strategies for dealing with the issues. The findings of the study point to a clear path for the development of federated learning systems for industrial use. Finally, this paper discusses future research trends in federated learning and encourages academics to build on their current work.

Data provenance protocols control the flow of data and information between nodes. They're good at preventing single-point-of-failure attacks and adversarial data tampering. Blockchains are one option for creating a data provenance mechanism in a federated learning system. For audit and data provenance, blockchains record all events rather than raw data, and only allow authorized parties access to the information. In addition, blockchains are used to store global model changes and incentive provisions in Merkle trees.

Data augmentation approaches are presented to address data security and client device security challenges. The approaches use privacy-preserving generative adversarial network (GAN) models to locally replicate data samples from all devices for data release and debugging operations. These approaches add an extra layer of security to safeguard the data itself from being exposed. Furthermore, by reproducing an IID dataset for superior training performance, data augmentation methods are effective in reducing statistical heterogeneity difficulties.

Auditing techniques are used to address the lack of system auditability and client device security constraints. They're in charge of assessing a client node's honest or semi-honest behavior during training and identifying any anomalies. Certain academics have also developed anomaly detection algorithms with the goal of penalizing hostile or misbehaving nodes.

Some academics have created evaluation tools to examine the behavior of a federated learning system more thoroughly. System auditability and statistical heterogeneity are addressed by the procedures. To demonstrate the federated learning system, for example, a visualization platform is available. A benchmarking platform is provided in order to accurately represent the characteristics of a federated environment.

The goal of this question is to identify the components of a federated learning system and their functions. There are two types of components in federated learning systems: central servers and client devices. A central server initiates and orchestrates the training process, while client devices perform the actual model training.

In some studies, in addition to the central server and client devices, an edge device was added to the system as an intermediate hardware layer between the central server and client devices. The edge device improves communication efficiency by reducing the training data sample size and increasing the update speed.

To determine where these components are stored, we classify them as client-based, server-based, or both. The model trainer is the most frequently mentioned component in client-based components, while the model aggregator is the most frequently mentioned component in server-based components. Anomaly detectors, data provenance, communication coordinators, resource managers, and client selectors, for example, are primarily server-based, whereas model performance software (such as feature fusion and data augmentation) is primarily client-based. Finally, on both clients and servers, two-way working software components such as model encryption, privacy protection, and model compression are available.

Central server: Local workstations, cloud servers, mobile edge computing platforms, edge gateways, and base stations are common locations for central servers. The central server is a hardware component that generates the first global model by randomly initializing the model parameters or gradient. Apart from randomizing the starting model, central servers can use a self-generated sample dataset or a small amount of data collected from each client device to pre-train the global model. The server-initialized model training configuration is what we call it. It's worth noting that not all federated learning systems use the central server to set up their global model. Clients create the global model in decentralized federated learning.

After the global model has been initialized, the central servers broadcast the global model, which contains the model parameters or gradients, to the participating client devices. Every round, the global model can be broadcast to all participating client devices or only to a subset of client devices, at random or based on model training results and resource availability. Similarly, all or a subset of the participating client devices are used to train local models. There are two ways to collect models: asynchronously and synchronously. After receiving all or a specified number of updates, the central server performs model aggregations before redistributing the revised global model to the client devices. This procedure is repeated until convergence has been achieved.

In addition to managing the exchange of model parameters and gradients, the central server hosts other software components such as encryption/decryption algorithms for model

encryption and decryption, and resource management mechanisms to optimize resource usage. The client and model selectors are described in order to select appropriate clients for model training and high-quality models for global aggregation, and a model and system performance assessment methodology is proposed. The use of feature fusion procedures to combine key model features while lowering communication costs is discussed. To persuade clients to participate, incentive strategies are used. The anomaly detector looks for anomalies in the system, while the model compressor compresses the model to save space. The communication coordinator is in charge of managing the multi-channel connection between the central server and client devices. Finally, using auditing tools, the training processes are audited.

Client Devices : Client devices are the hardware components that train models using datasets that are locally available. To begin, each client device gathers and pre-processes information from its immediate environment (data cleaning, labeling, feature extraction, etc.). The first global model is sent to all client devices, which starts the procedures. The global model parameters are decrypted and retrieved by the client devices. They then conduct model training on a local level. Client devices use the received global model for data inference and prediction.

The loss function is reduced and the local model's performance is improved with local model training. The model is usually trained for several rounds and epochs before being uploaded back to the central server for model aggregations. It was suggested that local training be done on numerous local mini-batch data sets to reduce the number of communication rounds. The technique only communicates with the central server once the model has reached convergence. The client devices then send the training results back to the central server (model parameters or gradients). Before uploading, client devices evaluate the performance of the local model, and only upload when the agreed-upon level of performance is met. Before uploading, the results are encrypted using the encryption process to ensure data security and prevent information leakage. In addition, the model is compressed before being sent to the central server to reduce connection costs. In some circumstances, not all devices are required to upload their results. Only selected client devices are required to upload the findings, depending on the selection criteria provided by the central server.

The criteria take into account the client devices' available resources as well as the model's performance. In the same federated learning systems, client devices can house data augmentation techniques and feature fusion processes that are linked to the central server.

When a model training round is finished, the training results are sent to a central server for global aggregation. Client devices in decentralized federated learning systems communicate with one another without requiring the intervention of a central server. As a software component for model and information provenance, the central server is mostly replaced by a blockchain. Incentives and differential private multiparty data model sharing are also handled by the blockchain. The initial model is created using local datasets on each client device. The models are updated using a consensus-based method that allows devices to communicate with one another and receive model updates and gradients from other nodes in the area. A peer-to-peer network is used to connect the client devices. A copy of the model update is installed on each client device. Once an agreement is reached, the revised gradient will be used to teach all client devices.

The system has a large client device population in cross-device settings, and each device owns its own data. The client network in a cross-silo environment, on the other hand, is set up by a number of businesses or organizations, regardless of how many individual devices they own. When compared to a cross-device setup, the number of data silos is significantly reduced. As a result, the cross-device system generates models for large-scale scattered data within the same application, whereas the cross-silo system generates models for data that differs in content and semantics in terms of both features and sample space. The data partitioning differs as well. The data is automatically partitioned by example for cross-device configuration. The cross-silo data splitting feature (vertical) or example-based data splitting is calculated (horizontal).

Mandatory components (clients) : Data collection, model training,preprocessing, feature engineering, and inference are all steps in the process.
Mandatory components (Server) : evaluation, model aggregation.
Optional components (clients) : Data augmentation, security protection, feature fusion/selection, privacy preservation, and data provenance, model compression, anomaly detection and auditing procedures.
Optional components (server) : Advanced model aggregation, incentive mechanism, resource management,training management and communication coordination.

2.2.4 Implementation and Evaluation

The federated learning systems, including the constructed models, must be evaluated after the architecture design stage and after the system has been implemented.

The focus of the question is on the study evaluation methods. We divide evaluation methods into two categories: simulation and case study. The most prevalent task for the simulation technique is image processing, whereas the most often used use cases are mobile applications such as word recommendation and human activity identification. Typically, researchers test their federated learning systems by simulating circumstances that are privacy sensitive. Google's mobile keyboard prediction is one of the few real-world instances.

By raising this topic, we want to find evaluation measures for both qualitative and quantitative approaches employed by federated learning systems. We explain how each evaluation statistic is used to assess the system and how these metrics are linked to the quality attributes. To begin, communication efficiency is measured using communication cost, dropout ratio, model performance, and system operating time. Comparing communication rounds to learning accuracy, the satisfying rate of communication overhead versus the number of clients, the theoretical analysis of communication cost of data interchange between the server and clients, data transmission rate, bandwidth, and communication latency, and the theoretical analysis of communication cost of data interchange between the server and clients are used to calculate the communication cost. To determine the dropout ratios, the computing overhead is compared to the dropout ratios. The results are presented as a communication overhead comparison for various dropout rates, as well as a performance comparison vs. dropout rate comparison.

Second, The model's performance is measured using the training loss, AUC-ROC value, F1-score, root-mean-squared error (RMSE), cross entropy, precision, recall, prediction error, mean absolute error, dice coefficient, and perplexity value. Third, the communication cost and system running time are used to assess the system's scalability. The system running time assessment displays the overall execution time of the training protocol (computation time and communication time), the running time of different operation phases, the running time of each round vs the number of client devices, and the model training time.

When evaluating the system's performance, security, scalability, and system reliability are all taken into account. The attack rate is calculated using the proportion of attack targets classified incorrectly as the target label. Researchers primarily use this statistic to evaluate the effectiveness of defense mechanisms. Examples of attacks include model poisoning, sybil, byzantine, and data reconstruction. Computation cost is a measurement of a system's compute, storage, and energy resource utilization. The computation overhead is determined by taking

into account the number of client devices, average calculation time, computation throughput, computation delay, computation utility, and component overhead. Storage resources are evaluated using memory and storage overhead, as well as storage capacity. Communication energy consumption, computing time energy consumption, and training dataset size energy consumption are all calculated using the following formulas. Finally, the convergence rate is calculated using the accuracy versus communication rounds, system running duration, and training data epochs.

Qualitative assessments are undertaken for statistical and system heterogeneity to see if the offered approaches have met their goal of addressing the limitations. The dropout ratio due to restricted resources and formal verification through equation proving are used to analyze statistical heterogeneity, whereas system heterogeneity is evaluated by the dropout ratio and formal verification through equation proving.

The incentive rate is used to determine how motivated a client is in regard to different aspects of the system. Calculating the task publisher's profit under various numbers of clients or degrees of accuracy, the average reward based on model performance, and the link between the offered reward rate and local accuracy over communication costs determines the incentive rate. In the research gathered, there is no mention of any specific form of reward offered as an incentive. Bitcoin and other cryptocurrencies, as well as tokens that can be traded for actual money, are common sorts of rewards.

Finally, attack rates and other qualitative evaluation criteria are used to assess both data security and client device security. A study of the performance of encryption and verification processes, differential privacy achievement, the impact of removing centralized trust, and the assurance of shared data quality assurance are among the data security studies. The performance of the encryption and verification procedure, the confidentiality guarantee for gradients, the auditability of gradient collection and update, and the fairness guarantee for model training are all included in the client device security evaluations. Furthermore, data security is measured by privacy loss, which uses differential average-case privacy to assess the suggested method's privacy-preserving level.

The federated learning system is evaluated using both quantitative and qualitative methods. Model performance, communication and computation costs, system running time, and so on are examples of quantitative measurements. Qualitative metrics include data security studies

on differential privacy attainment, encryption and verification method performance, gradient confidentiality guarantee, gradient collection and update auditability, and so on.

CHAPTER 3

HOMOMORPHIC ENCRYPTION

Homomorphic encryption is a form of encryption with an additional evaluation capability for computing over encrypted data without access to the private key . The result of such a computation remains encrypted but does computation over decrypted data. Homomorphic encryption can be viewed as an extension of public-key cryptography.

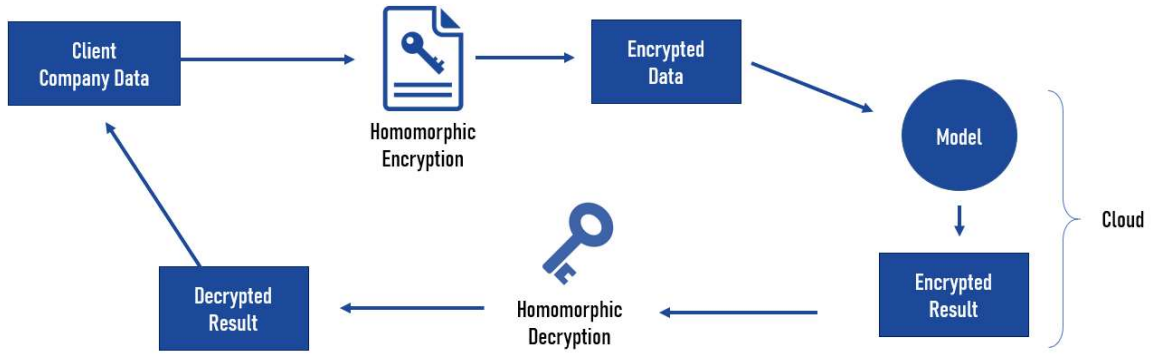


Fig 4: Homomorphic Encryption

3.1 Types of Homomorphic Encryption

Partially Homomorphic Encryption (PHE)

Any circuit composed of a single type of gate, addition or multiplication, but never both, can be evaluated using this method. It has no limitations on the size or depth of the circuit. This type is ideal for applications that only require the addition or multiplication of encrypted data. A PHE that allows an unbounded number of modular multiplications is the RSA cryptosystem.

This scheme allows the encryption values to be subjected to a single mathematical operation an unlimited number of times.

Somewhat Homomorphic Encryption (SHE)

This type of scheme can evaluate circuits that contain both addition and multiplication gates, but the depth is limited (e.g. circuits with a depth of at most 5). Leveled Homomorphic Encryption is a subset of SHE that can evaluate circuits with variable depth, but the depth must be set prior to encryption, so the parameters of your scheme must be tailored to the circuits you want to evaluate. SHE is useful for evaluating low degree polynomials up to a point, but we occasionally need to evaluate circuits of any depth.

This scheme, on the other hand, allows a limited set of operations to take place (more than one) but only a limited number of times.

Fully Homomorphic Encryption (FHE)

FHE schemes can evaluate circuits with both addition and multiplication gates, but unlike SHE, FHE has an unlimited circuit depth, making it ideal for deep learning. Despite the fact that numerous FHE schemes have been proposed over the last decade, implementing them has proven difficult. FHE is now based on SHE. Thanks to Craig Gentry for demonstrating in his paper how to bootstrap FHE from SHE.

This scheme allows any efficiently computed function (not just addition and multiplication) to be repeated an infinite number of times. Because of the latency, it loses its value despite being the most general form of HE. An FHE moves too slowly to be useful.

Among all these ,we used PHE to encrypt the model from the clients to the server .The server does computations on the decrypted data and the client only does computations on encrypted data.We used paillier encryption scheme to do encryption decryption in our project.

3.2 Deep Dive

Paillier Encryption

With public key $pk = n$, the encryption of an integer, the encryption of an integer $m \in 0, 1, 2 \ldots, n-1$ is $PaiEnc_{pk}(m) = r^n(1+n)^m$ in which r is chosen randomly from $\{0, 1, 2, \ldots n-1\}$ .The encryption is additively homomorphic as cryptoproduct $PaiEnc_{pk}(m1)PaiEnc_{pk}(m2) \mod n^2$ becomes an encryption $m_1 + m_2 \mod n$.

Packing data in Encryption

As plain text space has $log_2 n >= 2048$ bits, we can pack many integers, we can pack many integers $dat_1, dat_2 \ldots, dat_t$ into each pailler plain text as follows :

$$PaillerEnc_{pk}(\underbrace{dat_1 0pad}_{\text{prec+pad bits}} \ldots \underbrace{dat_t 0pad}_{\text{prec+pad bits}})$$

Here 0 pad is zero padding of pad bits which helps in overflows in cipher text additions. Moreover as number of plaintext bits must be less than $log_2 N_{data}$ it is necessary that

$$t(prec + pad) \leq log_2 N$$
$$\therefore t = log_2 N/(prec + pad)$$

which is the upper-bound of packing prec-bit integers into one Paillier plaintext.

As a real number 0 <= r < 1 can be represented as an integer of form $\lfloor r * 2^{prec} \rfloor$, the above packing method can be used to encrypt around $\therefore t = log_2 N/(prec + pad)$ bits.

Encryption cost

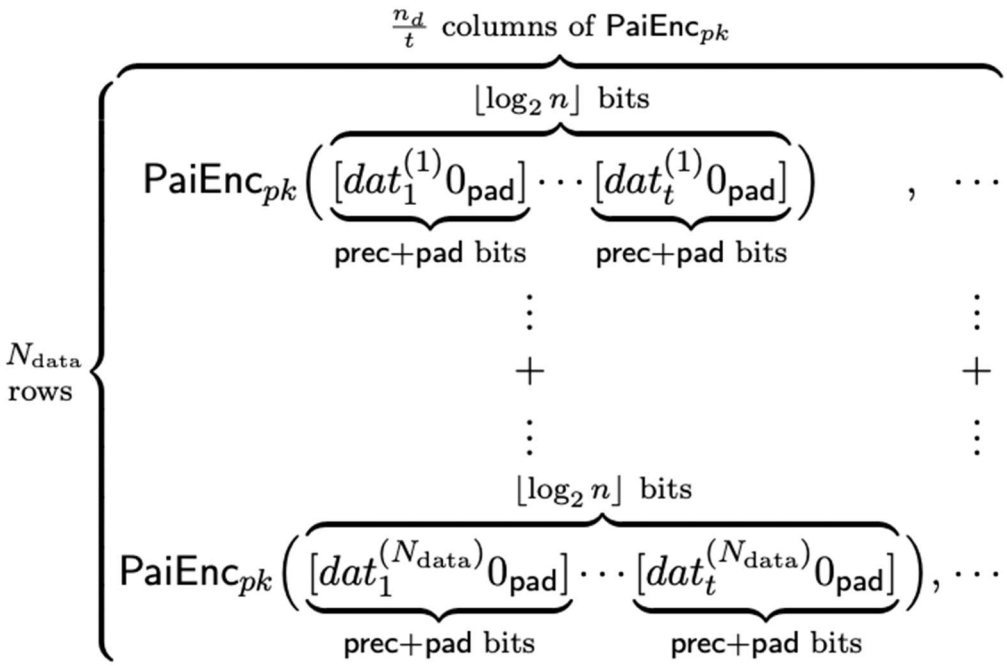
each data source need to encrypt and send the data to the server ,the cost would be around

$$ComunicationCostPailler = 2 * n_d * (prec + pad)$$

where $n_d = O(d^2)$ is the number of real numbers sent to the server.

Addition of ciphertexts

The ciphertext additions on the server can be seen as follows:



Whose computational cost is around

$$n_{data} \times n_d/t \times T_{PaiAdd}$$

Where $\boldsymbol{T_{PaiAdd}}$ is the time of adding two Paillier cipher texts.

The resulted n_d/t cipher text sums in columns are sent to the client, which also requires $O(2 \cdot d^2 \cdot (prec + pad))$ bits.

Decryption (client)

The client decrypts the n_d/t sums to obtain the sums $\sum_{i=1}^{N_{data}} dat_1^i \in \mathbb{Z} \ldots, \sum_{i=1}^{N_{data}} dat_t^i \in \mathbb{Z} \ldots, \sum_{i=1}^{N_{data}} dat_{n_d}^i \in \mathbb{Z}$.

As pad=$\lceil log_2 N_{data} \rceil$, there will be no overflows in integer sum as required.

CHAPTER 4

METHODOLOGIES AND IMPLEMENTATION

We reviewed several research papers and articles related to Federated Learning and then started experimentation in initial stages and implemented a few research papers.

A secure protocol for training on shared data has been used in previous studies. PHE is used by Liu et al [1] for distributed learning, in which each party sends encrypted weights to the server, which performs homomorphic additions and computes the total sum of weights. The total weights are then sent to all parties, starting with the leader party. To obtain a common dataset from different parties, Giacomelli et al [2] use Linear Homomorphic Encryption (LWE). This dataset is then used to train the Linear Regression model. This allows data to be exchanged in an encrypted format, which may not always be possible due to the privacy policies of many hospitals. In Aono et al [3], on the other hand, data is sent to the server in encrypted format, where it is processed using approximations and the model weights are then sent to the data owners. The encrypted data is shared between parties in this scenario, and the cost function approximation reduces the accuracy of the Logistic Regression model.

We train a Logistic Regression Classification model on the complete data of all the 'n' hospitals without accessing their databases or learning the actual data points because the task of classifying breast cancer into benign or malignant is binary. The hospitals' patient records will be different, but they will all have the same features. We divided our data into two sets in our problem statement: 'test' and 'train.'

For our 'n' number of hospitals, this train set is further divided into n shares. This illustrates the idea of a horizontal dataset split, i.e. different patients with the same feature set. All parties and experiments have access to the 'test' data.

Our approach is to share the derived information, such as gradients (used for gradient descent), in an encrypted format rather than the data in its true or encrypted form. The public-private key pair is created by company X, which then shares the public key with the hospitals for gradient encryption while keeping the private key to itself. We use the Paillier scheme (PHE) for key generation because addition is the only homomorphic computation we require.

Each hospital computes its own gradient, encrypts it with the public key, and then sends it to another hospital, which aggregates it to its own computed encrypted gradient, and so on.

The final hospital sends the total encrypted gradient to company X, which decrypts it and sends the total gradients back to all of the 'n' hospitals. The hospitals then use this aggregated gradient computed over the entire data to update their models.

Algorithm 1 Pseudo code

for hospital h_i in n **do**

$g_i \leftarrow$ compute gradient for h_i on data X_i

$ag_i \leftarrow g_i + ag_i - 1$ where aggregated gradient $ag_i - 1$ from $h_i - 1$

end for

dag ← Company decrypts ag

for hospital h_i in n **do**

h_i performs gradient descent using dag

end for

Model is trained on the whole data

This enables all hospitals to train their models on all sensitive data without having to share any information about the patients' personal information. Despite the fact that no data leaves the hospital, the model is trained on the entire dataset.

From the standpoint of security, we regard all parties as "honest but curious." Even if the true value of total gradient is known, no hospital will be able to pinpoint the source of a patient's data. This is true if at least three hospitals use this protocol, which prevents each other's gradients from being reconstructed by simply computing the difference. The total aggregated gradient also prevents the server 'X' from learning anything about the underlying data, protecting patients' privacy.

CHAPTER 5

RESULTS

5.1 Breast Cancer Dataset

We test all of the hospitals using the same data. None of the hospitals had previously seen any of the data points in the test data. We compare the performance of models that are trained on their own databases to the performance of models that are trained on the entire dataset, as seen earlier.

The experiments are carried out in a variety of hospitals. We've noticed that as the number of hospitals grows, so does the average accuracy of those hospitals (Table 1). On the entire plaintext training data, we also compare the results with scikit-inbuilt learn's Logistic Regression (Table 1).

We used a key length of 1024 bits in all of the experiments.

Table 1: Comparison of our model's performance on breast cancer data with the one without combining data and also, against scikit-learn's Logistic Regression. The best performances are indicated in bold. LL-Local Learning ; FL-Federated Learning

Number of hospitals (n)	Avg Acc (LL) (in %)	Train Time (LL) (in s)	Avg Acc (FL) (in %)	Train Time (FL) (in s)	Avg Acc (sklearn) (in %)	Train Time (sklearn) (in s)
3	**96.27**	0.003	95.80	2.821	95.80	0.123
7	**96.00**	0.006	95.80	6.315	95.80	0.028
10	95.38	0.008	**95.80**	9.987	**95.80**	0.033
20	94.93	0.161	**95.80**	18.343	**95.80**	0.036
27	94.17	0.209	**95.80**	24.681	**95.80**	0.038

5.2 Grad Admission DataSet

We can also see that the increase in performance comes at the expense of training time, but the goal remains the same: to create a secure model capable of higher accuracy. The sklearn Logistic Regression model performs similarly to ours, but it is faster because it works with plaintext data. It's possible that the difference in accuracy for lower 'n' values is due to the dataset. As a result, we compare our model's performance on the grad-admissions dataset to that of a local or independent learning model (Table 2).

Table 2: Comparison of our model's performance on grad-admission data with the one without combining data and also, against scikit-learn's Logistic Regression

Number of Data Owners (n)	Avg Acc (LL) (in %)	Train Time (LL) (in s)	Avg Acc (FL) (in %)	Train Time (FL) (in s)	Avg Acc (sklearn) (in %)	Train Time (sklearn) (in s)
3	70.40	0.003	**86.40**	0.777	**86.40**	0.024
7	80.46	0.006	**86.40**	1.709	**86.40**	0.029
10	79.28	0.014	**86.40**	2.594	**86.40**	0.027
20	75.24	0.017	**86.40**	5.143	**86.40**	0.030
27	77.16	0.020	**86.40**	6.802	**86.40**	0.024

Hence, we can conclude that our model will outperform the local learning model and will give as good an accuracy as the scikit-learn model.

5.3 With Acute Inflammation Dataset

The main idea of this data set is to prepare the algorithm of the expert system, which will perform the presumptive diagnosis of two diseases of the urinary system. It will be the example of diagnosing the acute inflammations of urinary bladder and acute nephritis. For better understanding of the problem, let us consider definitions of both diseases given by medics.

Acute inflammation of the urinary bladder is characterized by sudden occurrence of pains in the abdomen region and the urination in the form of constant urine pushing, micturition pains and sometimes lack of urine keeping. Temperature of the body is rising, however most often not above 38C. The excreted urine is turbid and sometimes bloody. At proper treatment, symptoms decay usually within several days. However, there is an inclination to return. For persons with acute inflammation of the urinary bladder, we should expect that the illness will turn into protracted form.

Acute nephritis of renal pelvis origin occurs considerably more often at women than at men. It begins with sudden fever, which reaches, and sometimes exceeds 40C. The fever is accompanied by shivers and one- or both-side lumbar pains, which are sometimes very strong. Symptoms of acute inflammation of the urinary bladder appear very often. Quite not infrequently there is nausea and vomiting and spread pains of the whole abdomen.

Attribute Information:

There are 6 features and 2 diagnoses (8 columns):

- **a1** Temperature of patient { 35C-42C }
- **a2** Occurrence of nausea { yes, no }
- **a3** Lumbar pain { yes, no }
- **a4** Urine pushing (continuous need for urination) { yes, no }
- **a5** Micturition pains { yes, no }
- **a6** Burning of urethra, itch, swelling of urethra outlet { yes, no }

- **d1** decision: Inflammation of urinary bladder { yes, no }
- **d2** decision: Nephritis of renal pelvis origin { yes, no }

There are ‘n’ hospitals. (The dataset will be split in n, randomly.) There could be more hospitals. The n hospitals cannot share the cases of their patients because they are competitors and it is necessary to protect the privacy of patients. Hence, the ML model will be learned in a federated way.

How? Federated learning is iterated 1000 times. At each iteration, a copy of the shared model is sent to all the 4 hospitals. Each hospital trains its own local model with its own local dataset, in 5 local iterations. Each local model improves a little bit in its own direction. Then we compute the local losses and local accuracies to keep track of them and to make graphs of them. We send the local models to the trusted aggregator that will average all the model updates. This averaged model is the shared model that is sent to all the 4 hospitals at the beginning of each iteration.

In this way, only the ML model will be shared. Whereas the local cases of each hospital will be kept private and they will be used to train model updates in a local way. Federated learning will protect the privacy of datasets in each hospital and at the same time, we will generate a more robust machine learning model, which will benefit all hospitals. This shared ML model preserves the privacy of individual patients and at the same time, reveals important statistics of stereotypical cases.

Here if we choose the n =4, we get the following results:

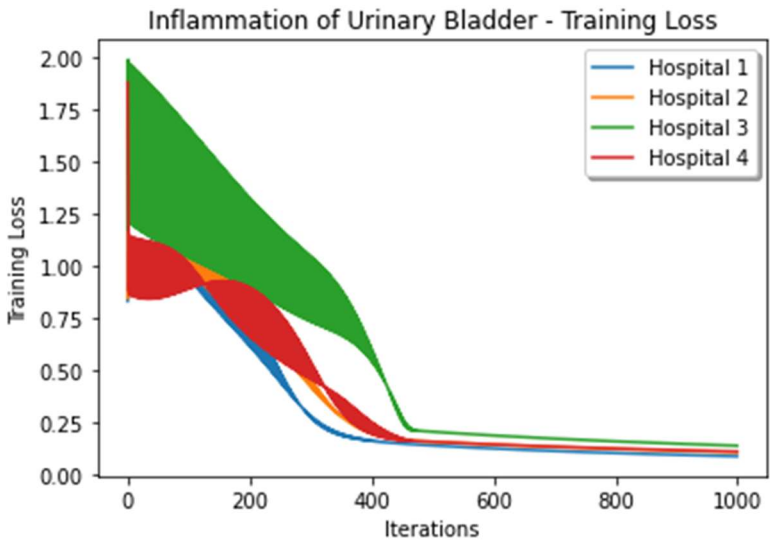


Fig 5: Inflammation of Urinary Bladder -Training Loss

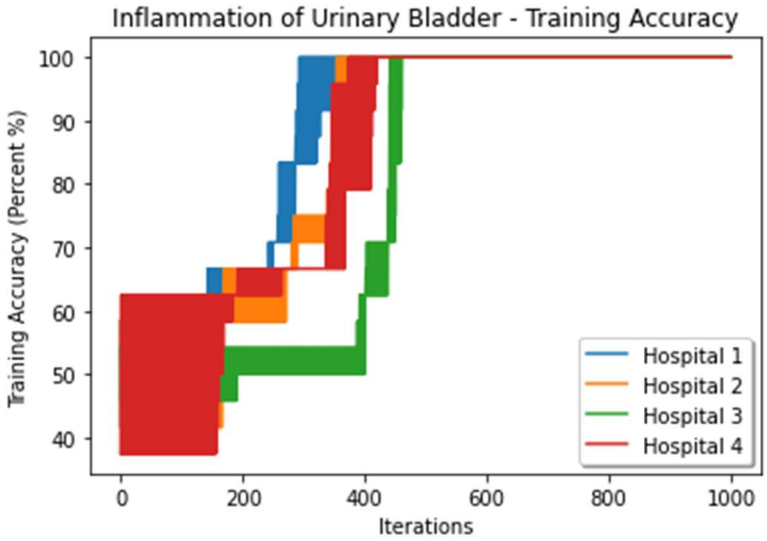


Fig 6: Inflammation of Urinary Bladder - Training Accuracy

5.4 Fed-Avg and Fed-Prox with MNIST dataset

Dataset : Load the MNIST Train and Test Dataset from the Torchvision Datasets

Partition : IId and Non-IId

First we defined two functions to partition the data Analogous to the real world scenario

1. IID : Identical and independent Dataset (Uniform Distribution)
2. Non-IId : it is a pathological implementation here every client in the FL system gets data only from a subset of classes but not all.

Model :

MNIST 2NN: Multi-Layer Perceptron replicated from the FedAvg paper.

MNIST CNN: Two layer CNN model replicated from the FedAvg paper.

Testing Loop : We use this test loop at every epoch to stop our model from overfitting.

FedProx :

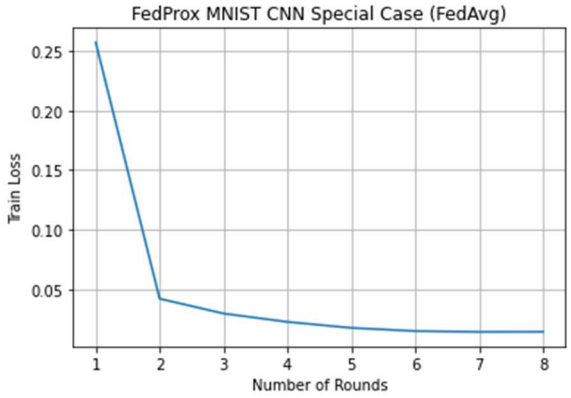
This block contains the entire implementation of the FedProx algorithm as discussed in the Fed-Prox Research paper. Different functions are implemented to replicate heterogeneity as mentioned in the FedProx paper. Client Update function is used to train local clients. Server Side Update function takes the local models by clients and updates the global model.

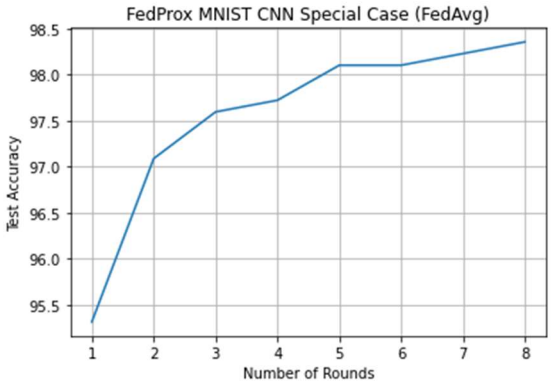
NOTE: FROM HERE WE START RUNNING DIFFERENT TESTS ON FEDPROX ALGORITHM

For each experiment, we played around the hyperparameters.

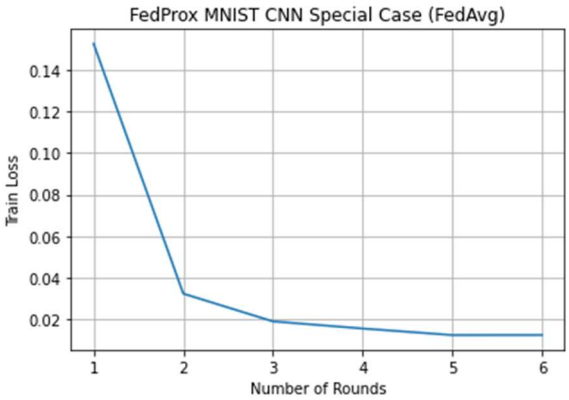
5.4.1 Special Case for FedProx : mu = 0 (Fed-Prox Act Like Fed-Avg)

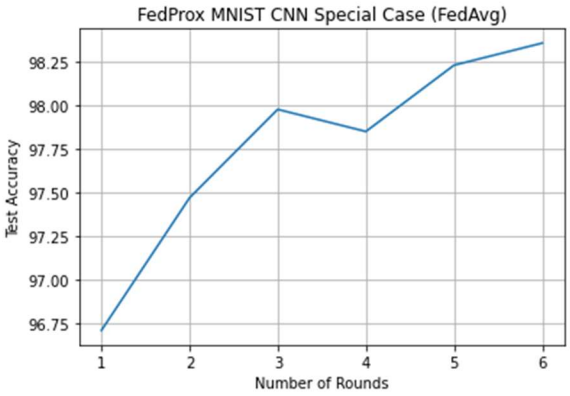
1. Learning-rate = 0.01, Round_taken = 8, heterogeneity = 0%,Test_Loss = 0.06, Test_accuracy = 98.35, IId



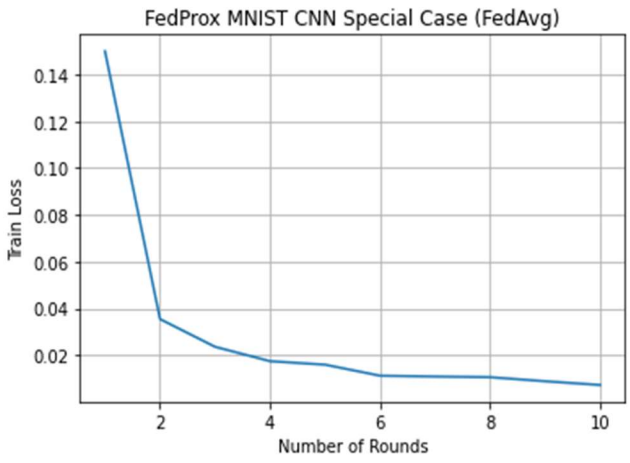


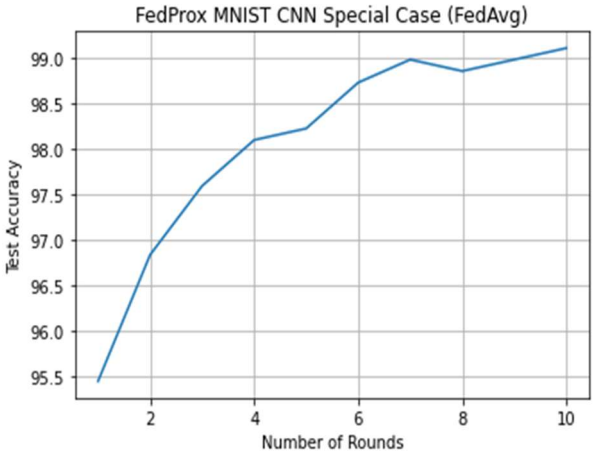
2. Learnin_rate = 0.03,Round_taken = 6, heterogeneity = 0%,loss = 0.06,accuracy = 98.35, IId





3. Learnin_rate = 0.05,Round_taken = 10, heterogeneity = 0%,loss = 0.039,accuracy = 99.11,





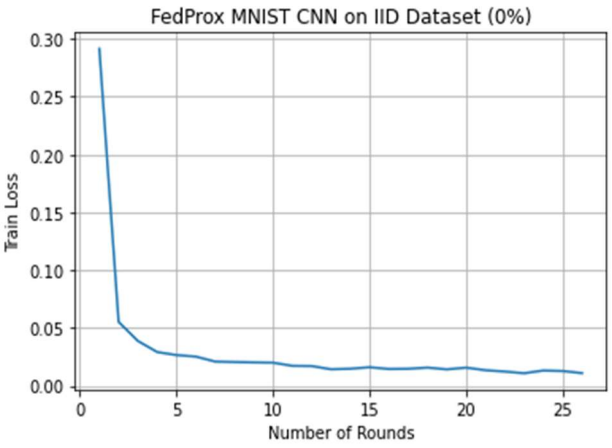
IId

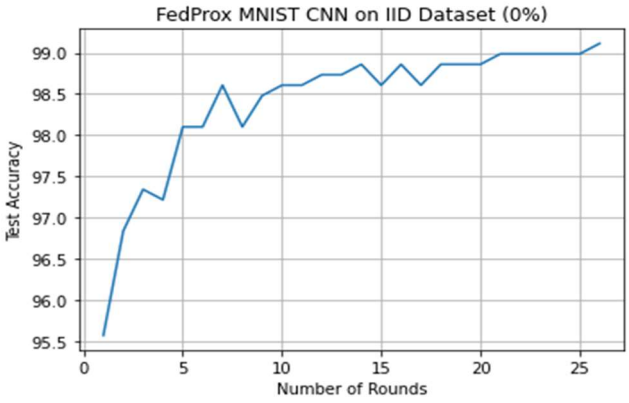
5.4.2 : Training with 0% Probability of Stragglers

CNN and MLP models for IID and non-IID data set when there are no stragglers.

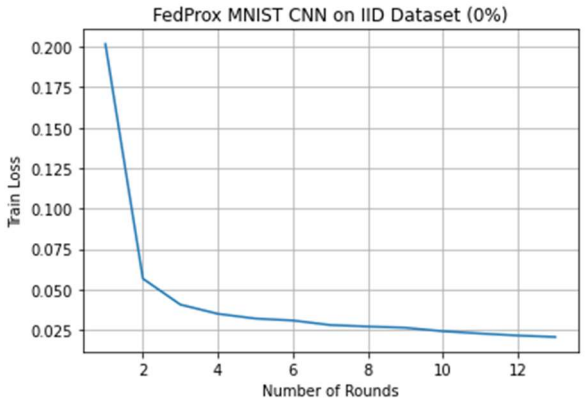
1. CNN on IId (0 %) :

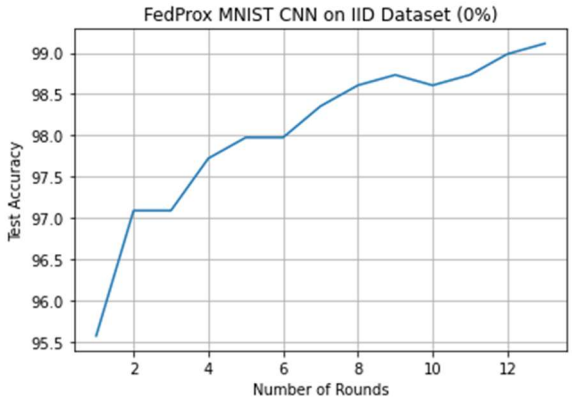
(a) lr= 0.01, round=26, loss = 0.04,accuracy = 99.11



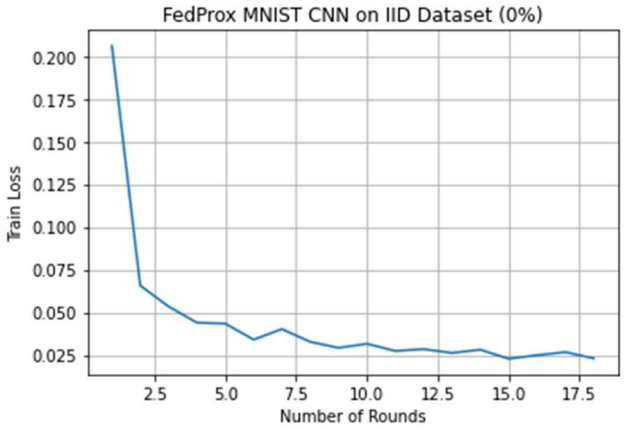


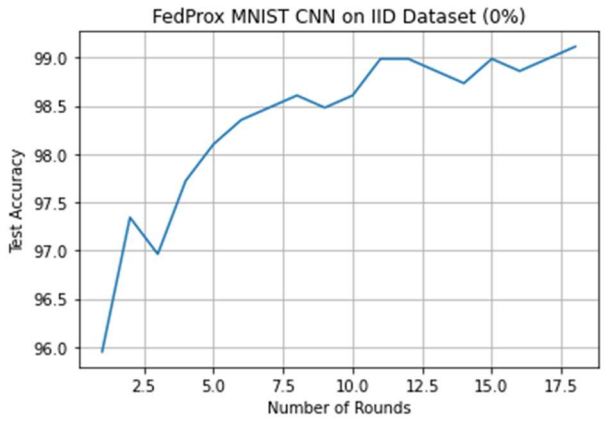
(b) lr= 0.03, round=13, loss = 0.05,accuracy = 99.11





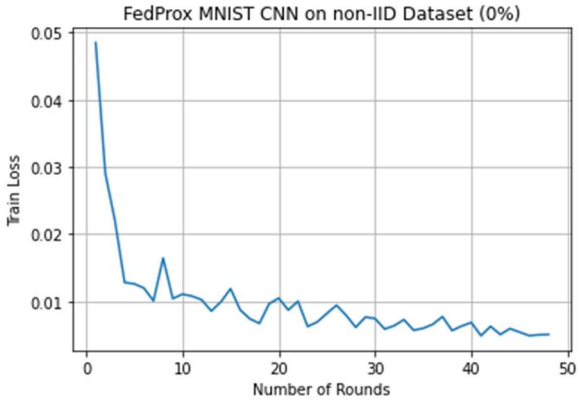
(c) lr= 0.05, round=18, loss = 0.038,accuracy = 99.11

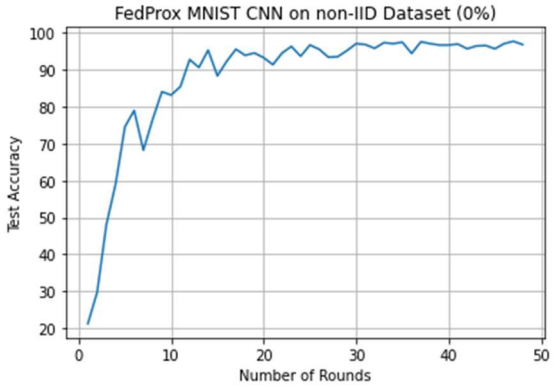




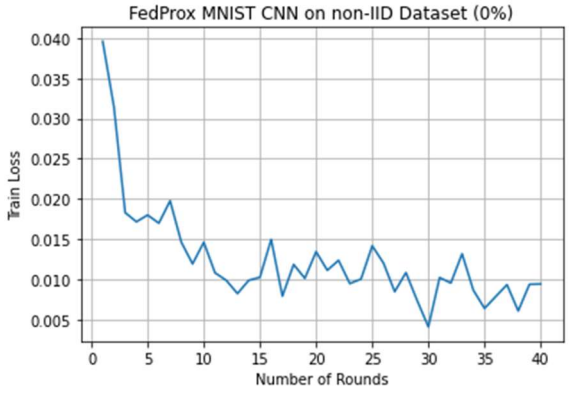
2. CNN on Non-iid (0 %) :

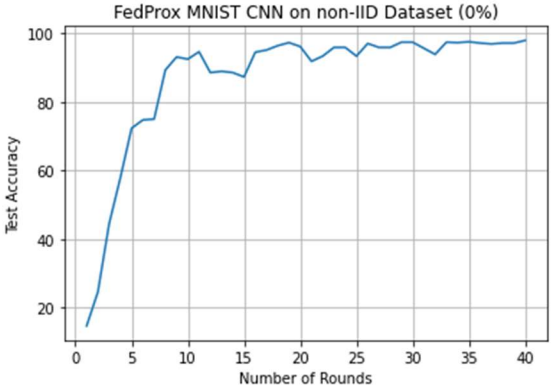
(a) lr= 0.01, round=48, loss = 0.11,accuracy = 96.83



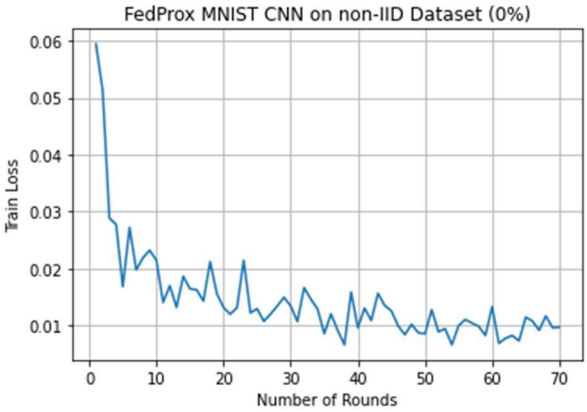


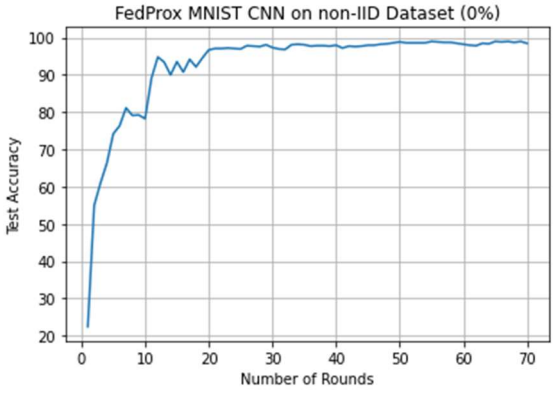
(b) lr= 0.03, round=40, loss = 0.07,accuracy = 97.84





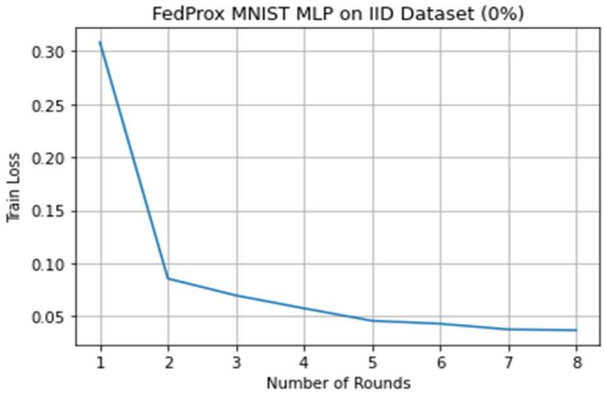
(c) lr= 0.05, round=70, loss = 0.04,accuracy = 98.48

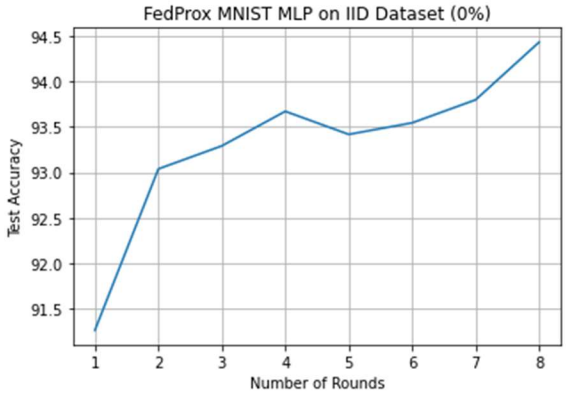




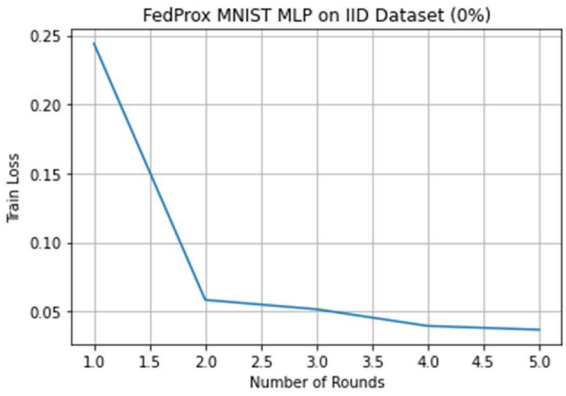
3. MLP on IId (0 %) :

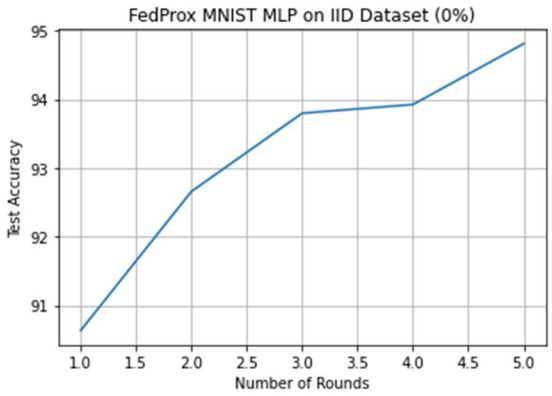
(a) lr= 0.01, round=8, loss = 0.20,accuracy = 94.43



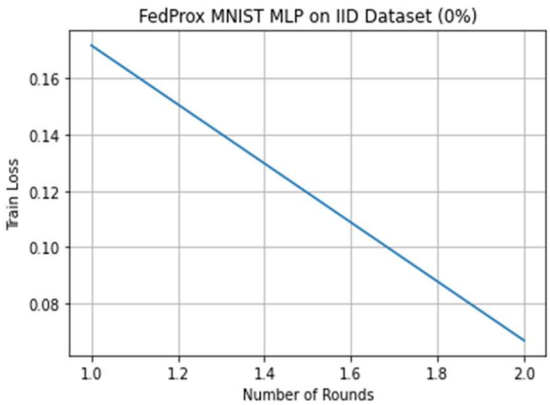


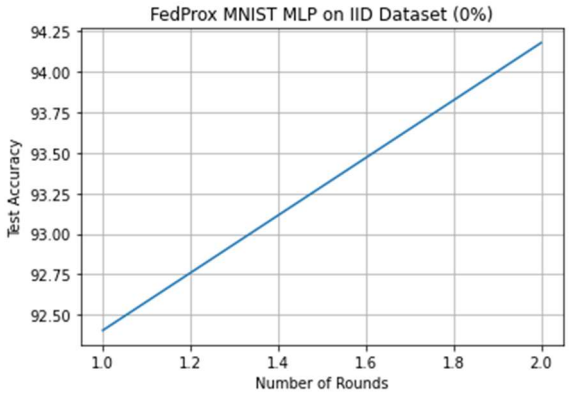
(b) lr= 0.03, round=5, loss = 0.22,accuracy = 94.81





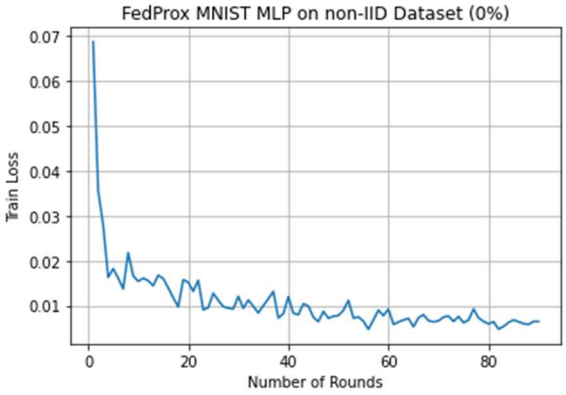
(c) lr= 0.05, round=2, loss = 0.28,accuracy = 94.17

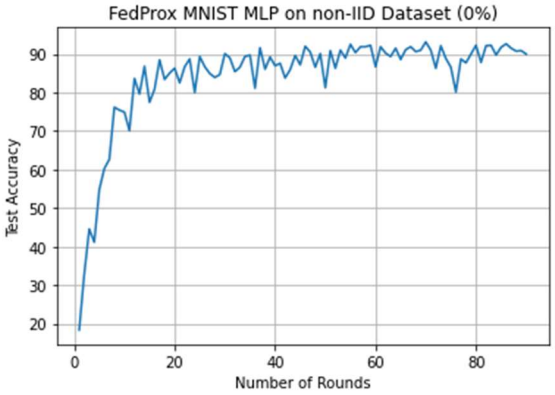




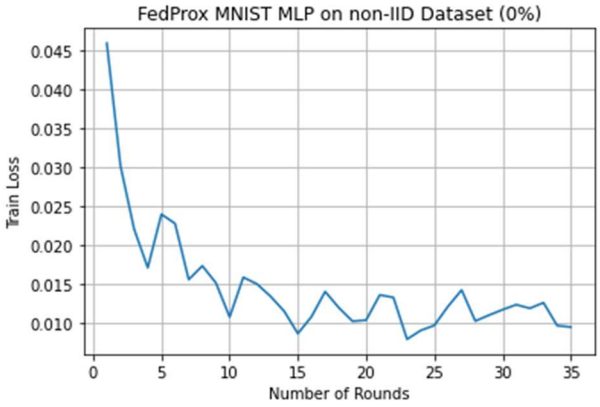
4. MLP on Non-IId (0 %) :

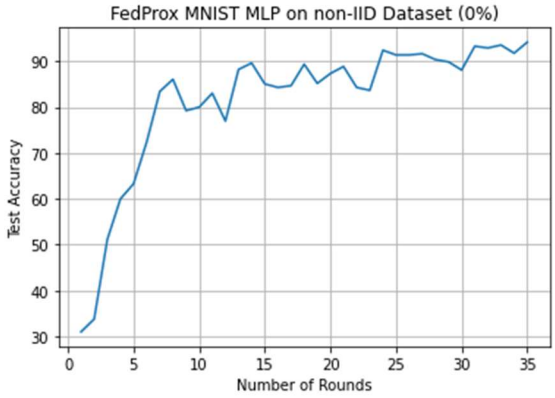
a. lr= 0.01, round=90, loss = 0.28,accuracy = 90.00



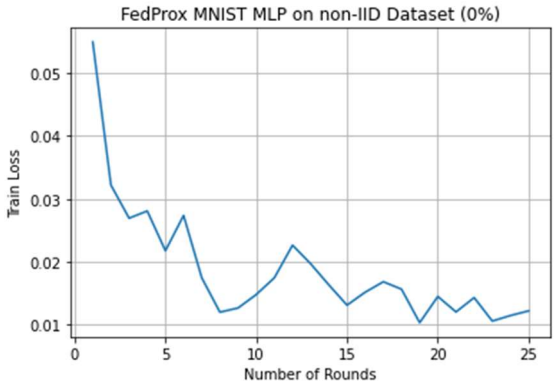


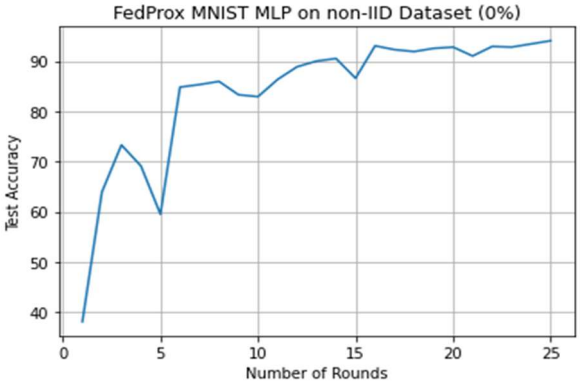
b. lr= 0.03, round=35, loss = 0.20,accuracy = 94.17





c. lr= 0.05, round=25, loss = 0.21,accuracy = 94.05



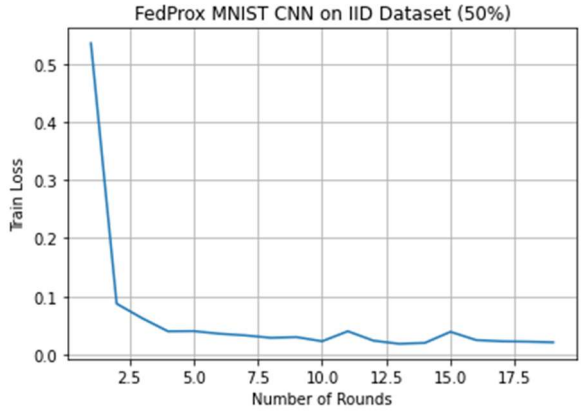


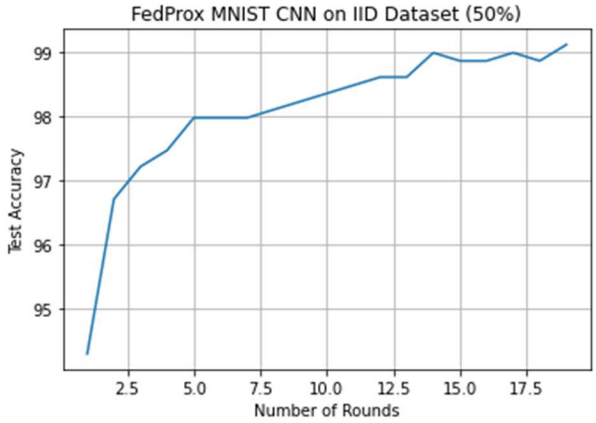
5.4.3 Training with 50% Probability :

CNN and MLP models for IID and non-IID data set when there are 50% stragglers.

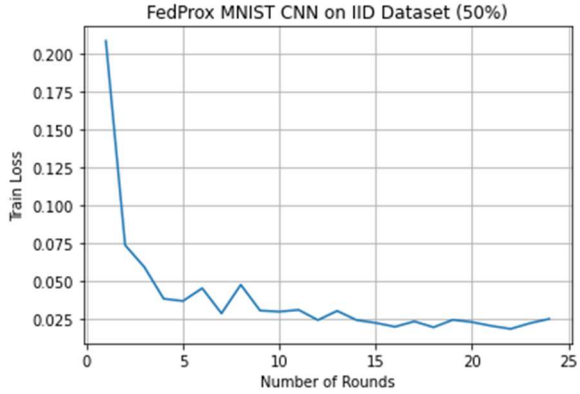
1. CNN on IId (50 %)

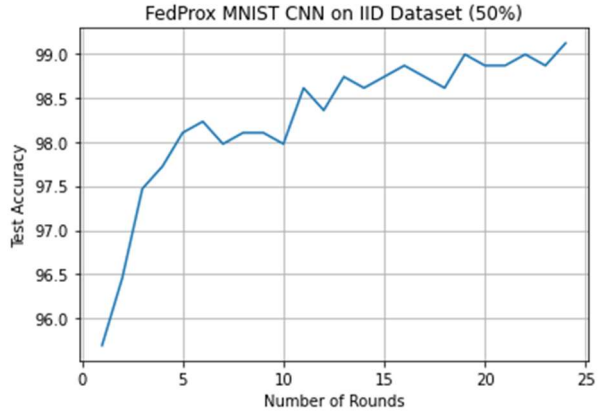
a. lr= 0.01, round=19, loss = 0.042,accuracy = 99.11



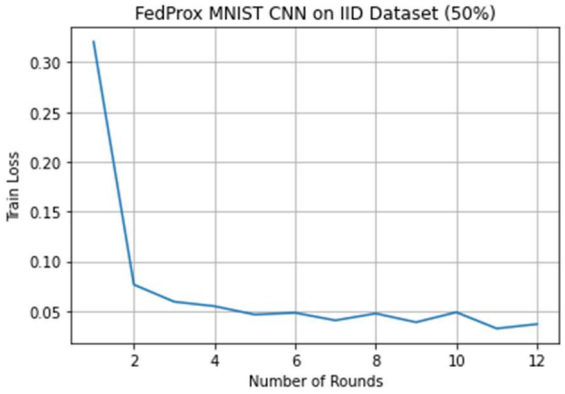


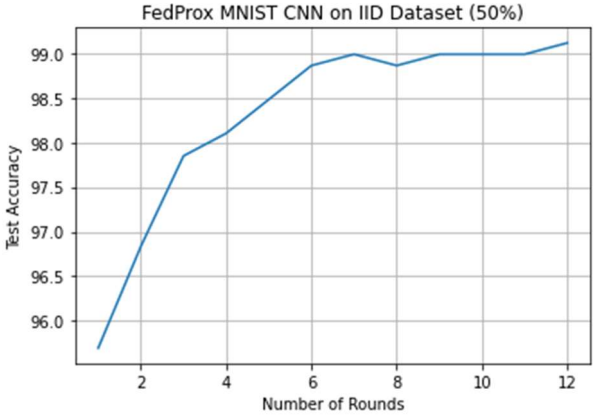
b. lr= 0.03, round=24, loss = 0.036,accuracy = 99.11





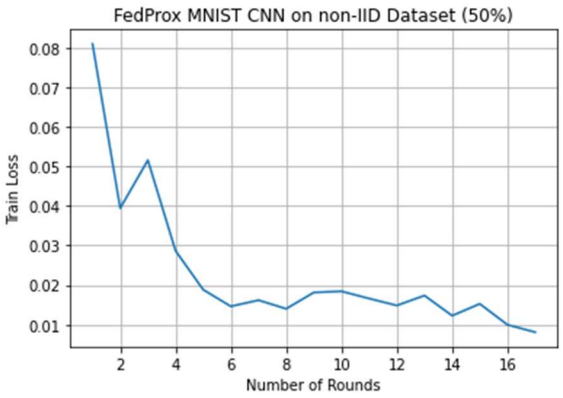
c. lr= 0.05, round=12, loss = 0.047,accuracy = 99.11

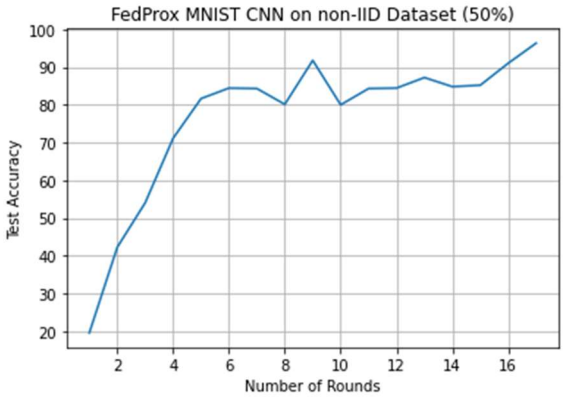




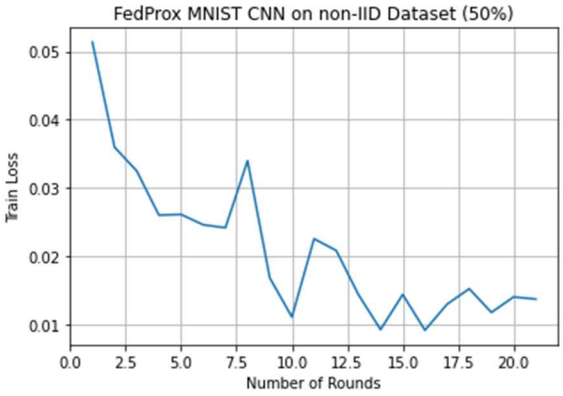
2. CNN on Non-IId (50 %)

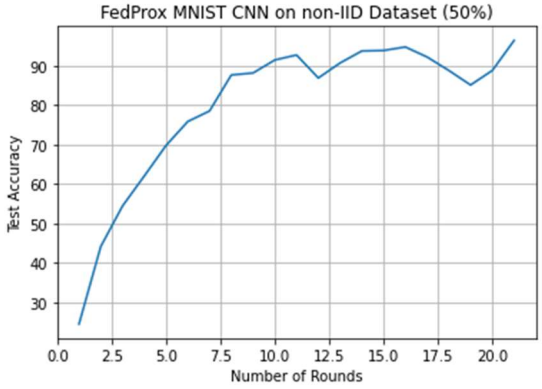
a. lr= 0.01, round=17, loss = 0.13,accuracy = 96.32



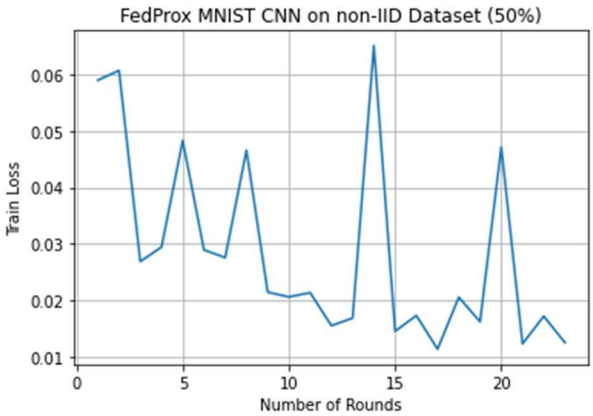


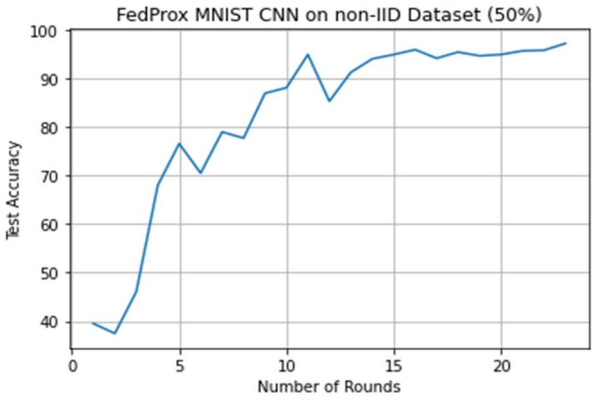
b. lr= 0.03, round=21, loss = 0.12,accuracy = 96.32





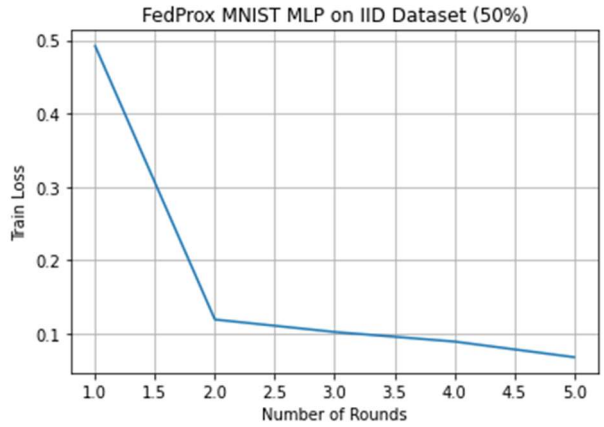
c. lr= 0.05, round=23, loss = 0.09,accuracy = 97.21

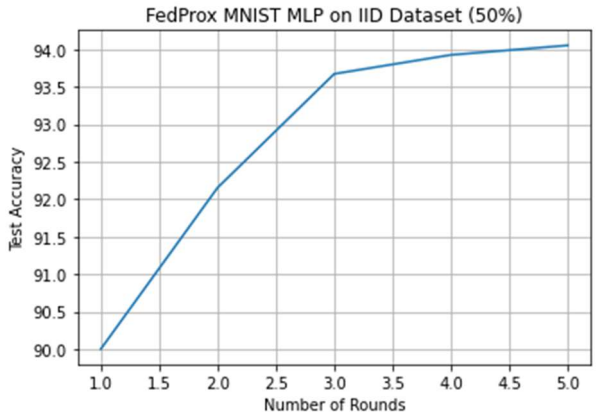




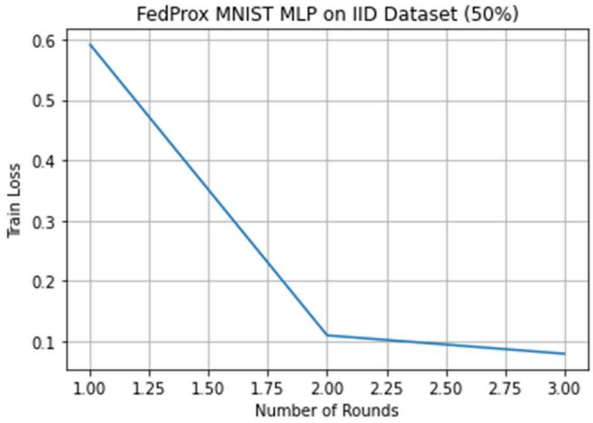
3. MLP on IId (50 %) :

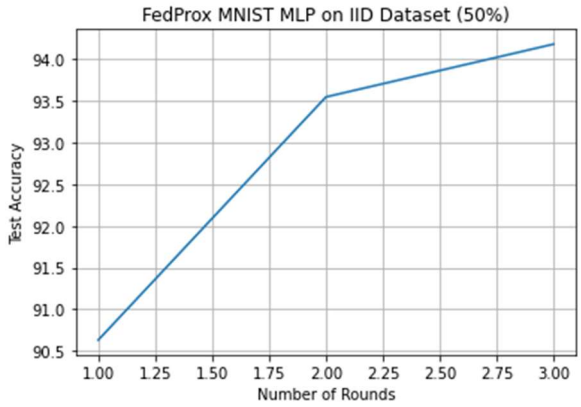
a. lr= 0.01, round=5, loss = 0.25,accuracy = 94.05



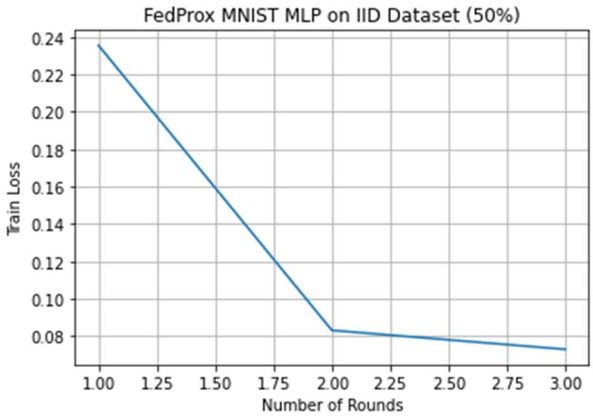


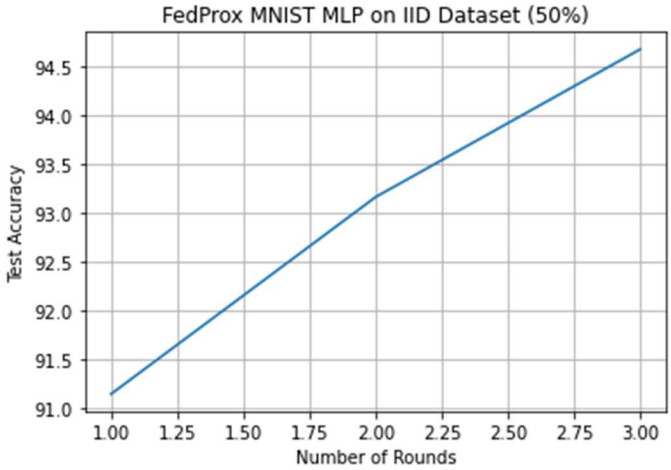
b. lr= 0.03, round=3, loss = 0.25,accuracy = 94.17





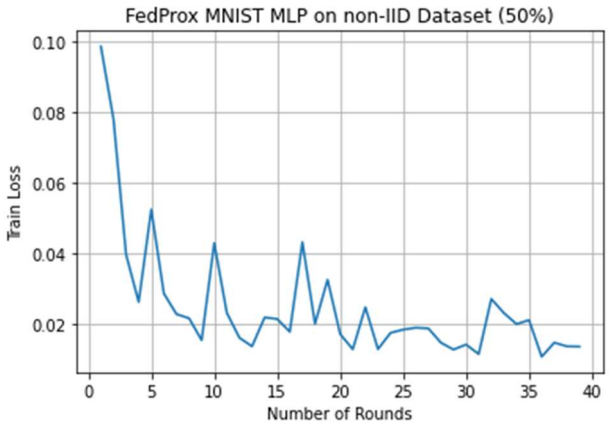
c. lr= 0.05, round=3, loss = 0.24,accuracy = 94.68

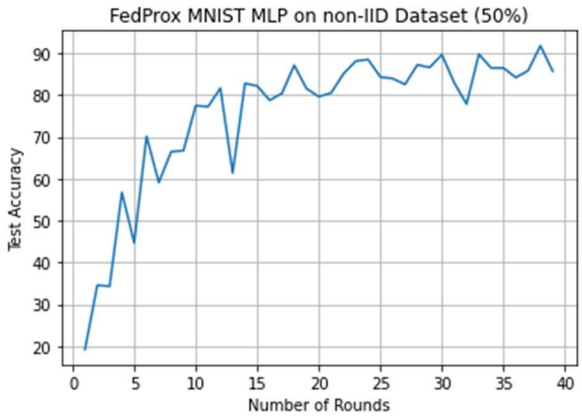




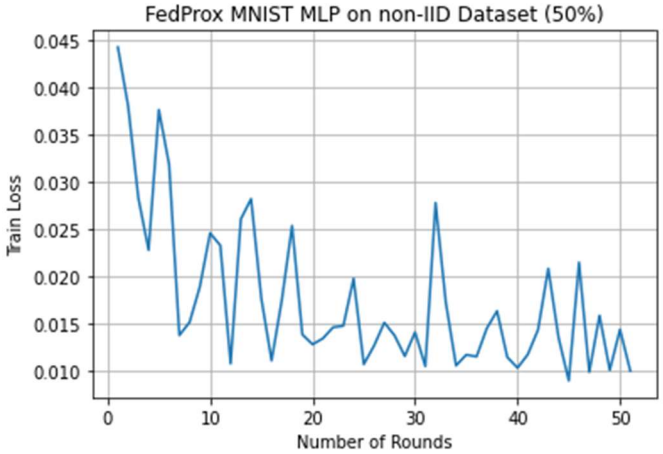
4. MLP on Non-IId (50 %)

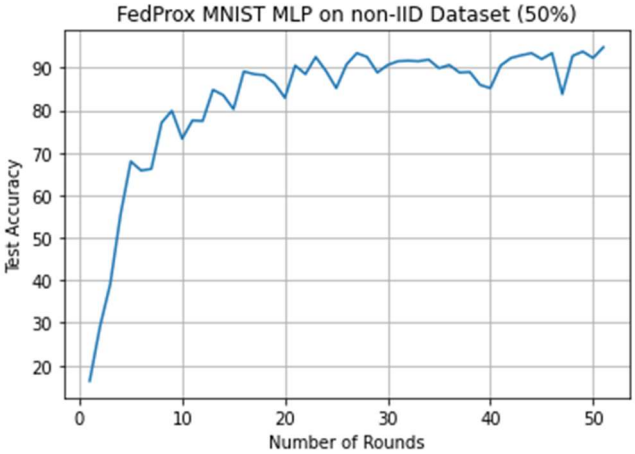
a. lr= 0.01, round=39, loss = 0.38,accuracy = 85.69



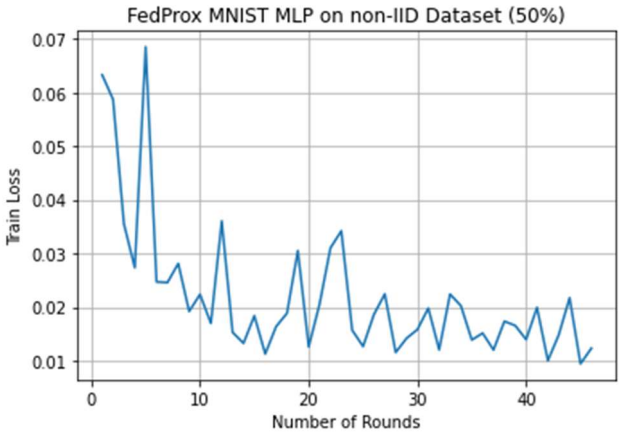


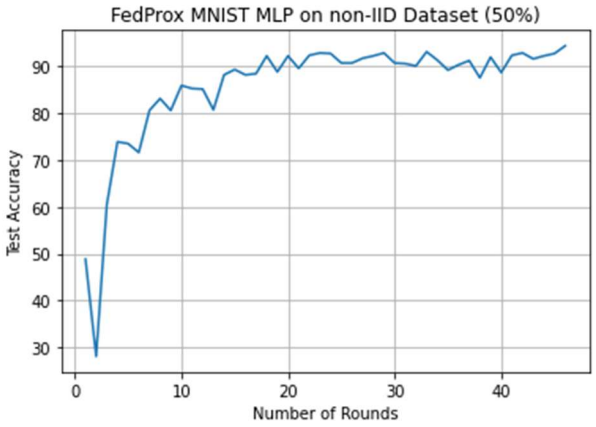
b. lr= 0.03, round=51, loss = 0.20,accuracy = 94.81





c. lr= 0.05, round=46, loss = 0.16,accuracy = 94.43



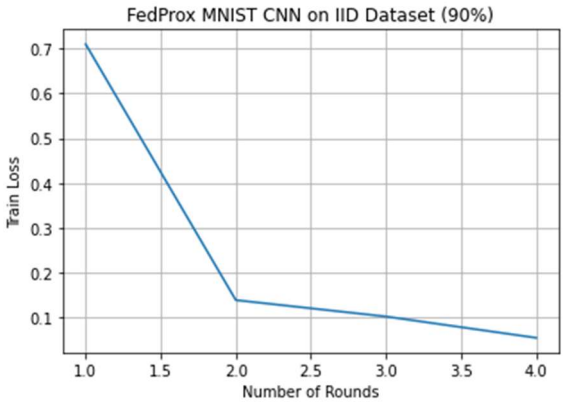


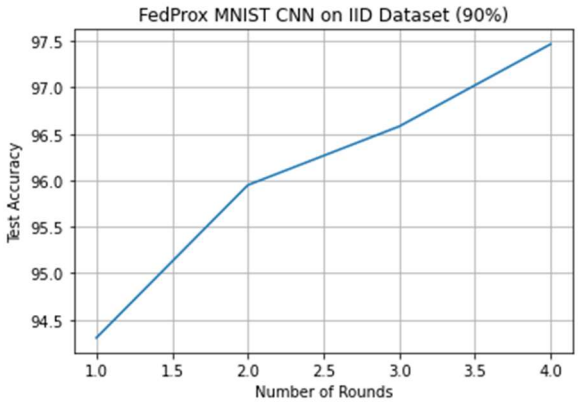
5.4.4 Training with 90% Probability

CNN and MLP models for IID and non-IID data set when there are 90% stragglers.

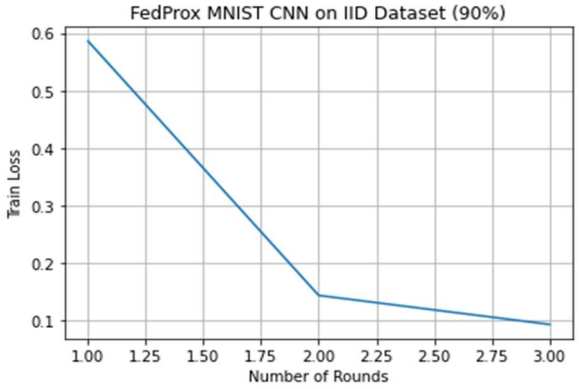
1. CNN on IId (90 %)

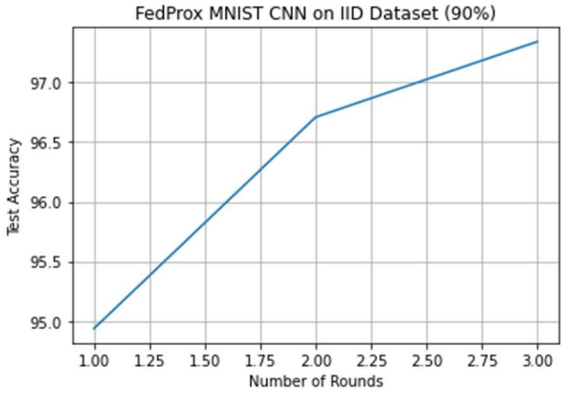
a. lr= 0.01, round=4, loss = 0.09,accuracy = 97.46



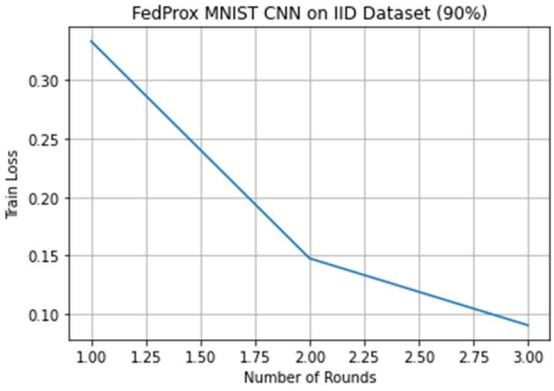


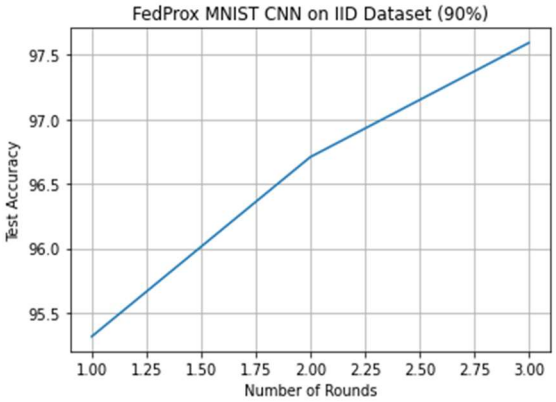
b. lr= 0.03, round=3, loss = 0.09,accuracy = 97.34





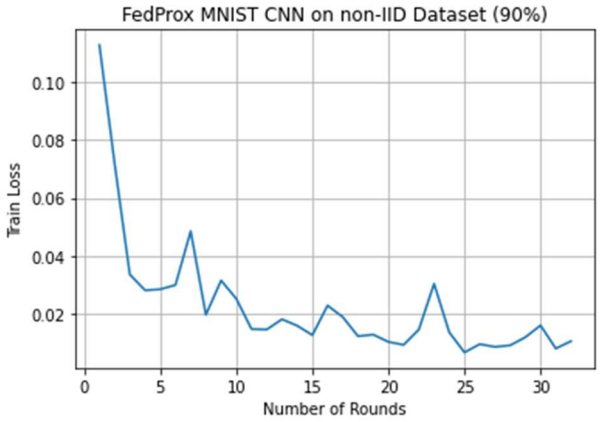
c. lr= 0.05, round=3, loss = 0.08,accuracy = 97.59

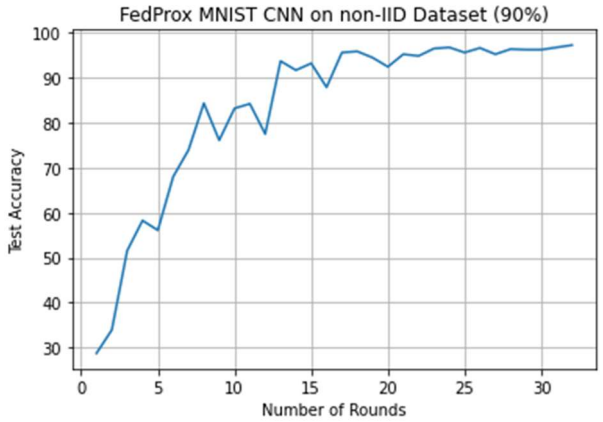




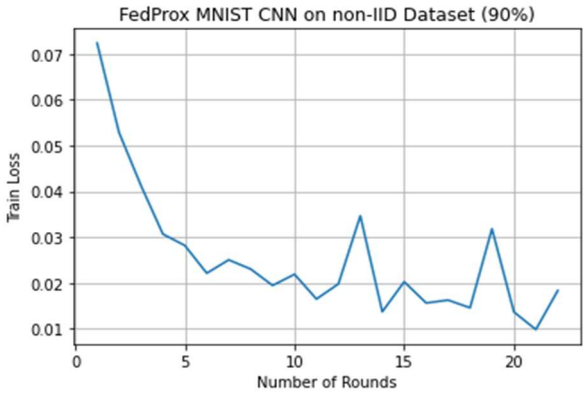
2. CNN on Non-IId (90 %)

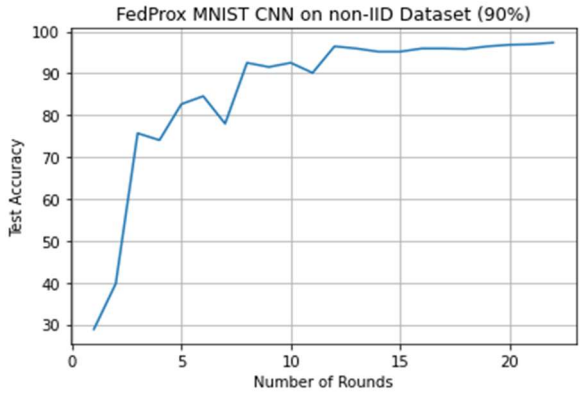
a. lr= 0.01, round=32, loss = 0.10,accuracy = 97.21



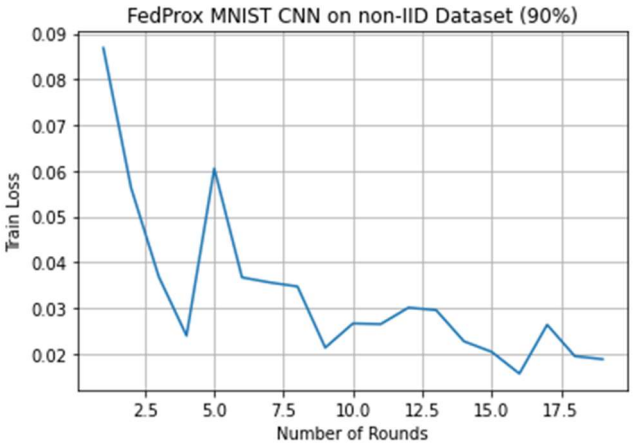


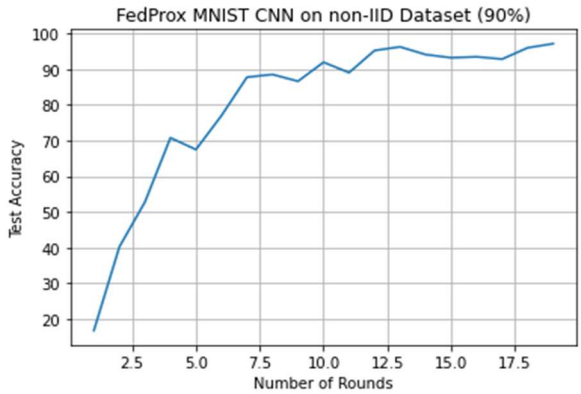
b. lr= 0.03, round=22, loss = 0.10,accuracy = 97.34





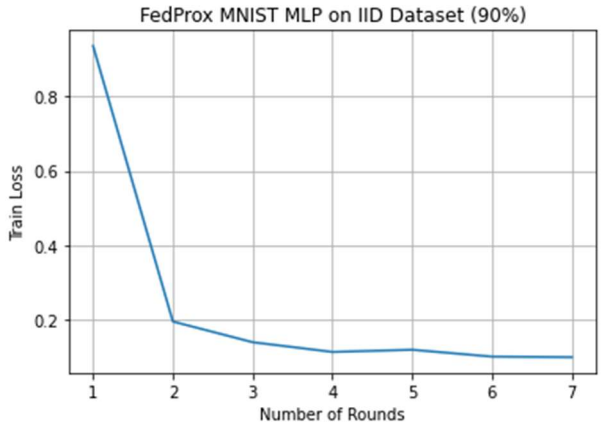
c. lr= 0.05, round=19, loss = 0.10,accuracy = 97.08

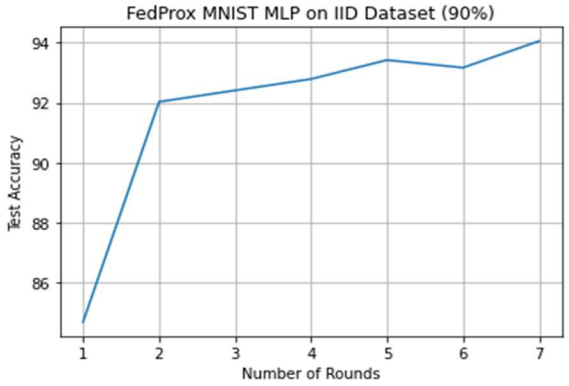




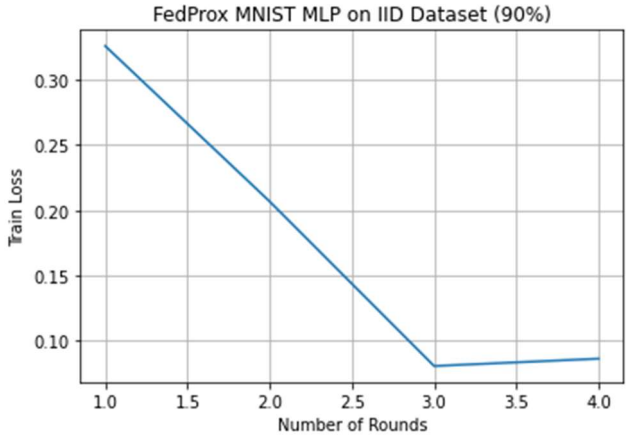
3. MLP on IId (90 %)

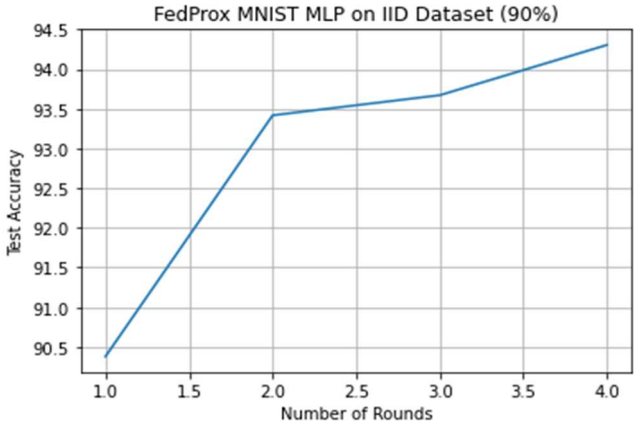
a. lr= 0.01, round=7, loss = 0.22,accuracy = 94.05



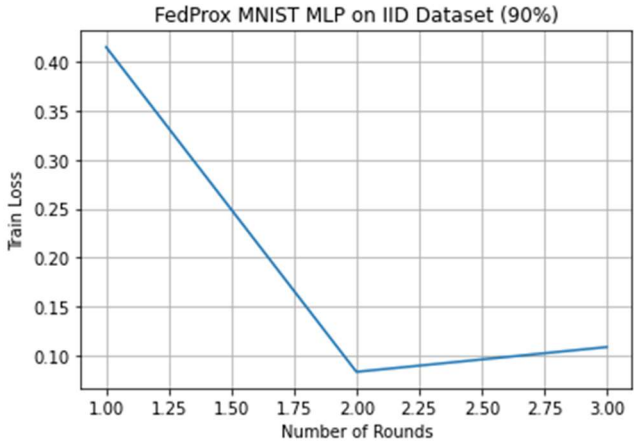


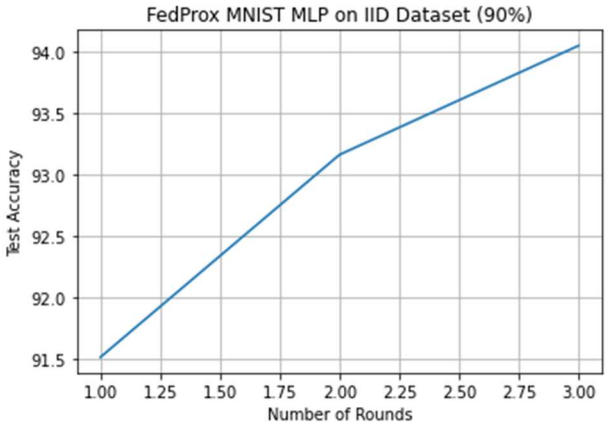
b. lr= 0.03, round=4, loss = 0.23,accuracy = 94.30





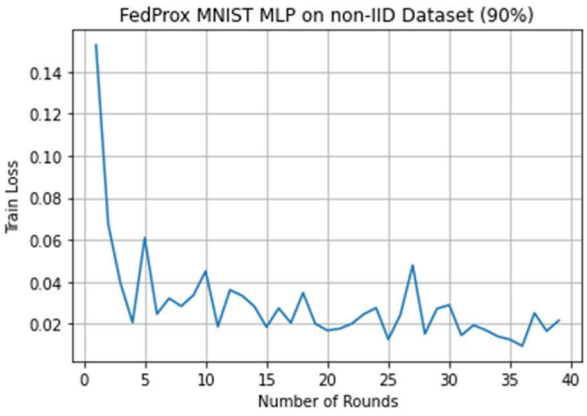
c. lr= 0.05, round=3, loss = 0.23,accuracy = 94.05

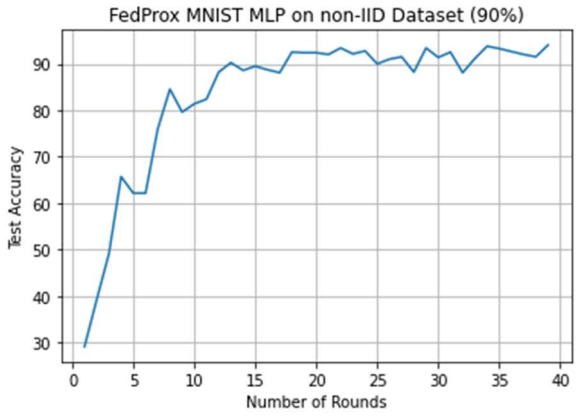




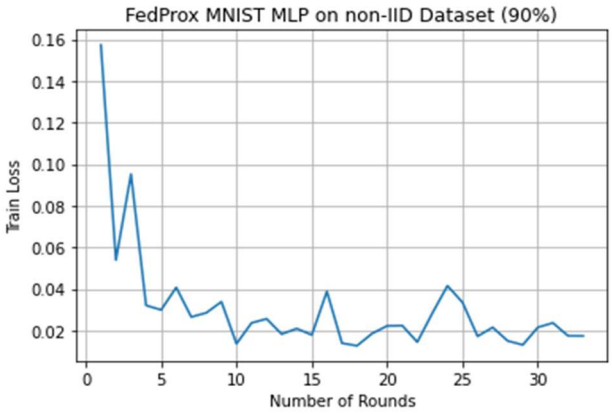
4. MLP on Non-IId (90 %)

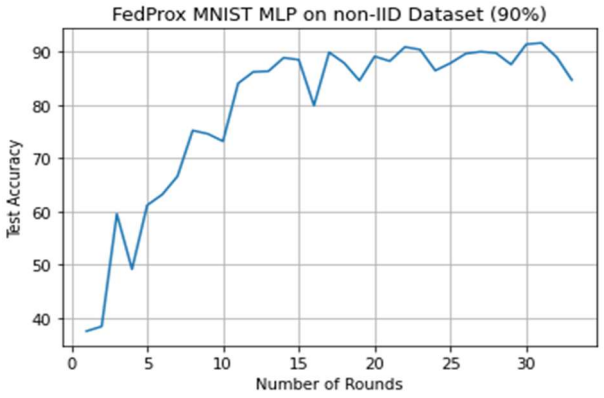
a. lr= 0.01, round=39, loss = 0.18,accuracy = 94.05



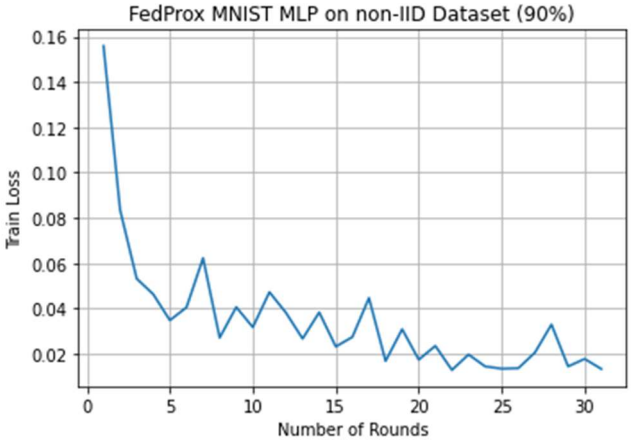


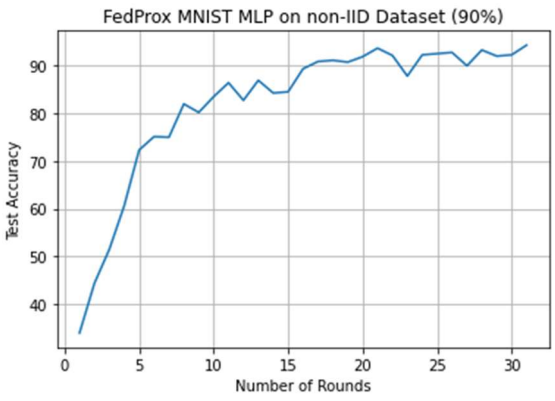
b. lr= 0.03, round=33, loss = 0.40,accuracy = 84.68





c. lr= 0.05, round=31, loss = 0.21,accuracy = 94.17

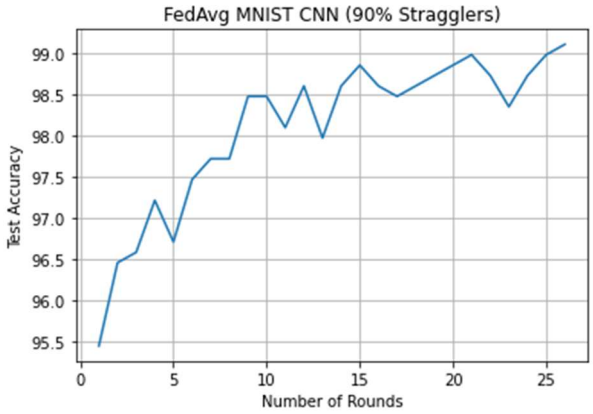


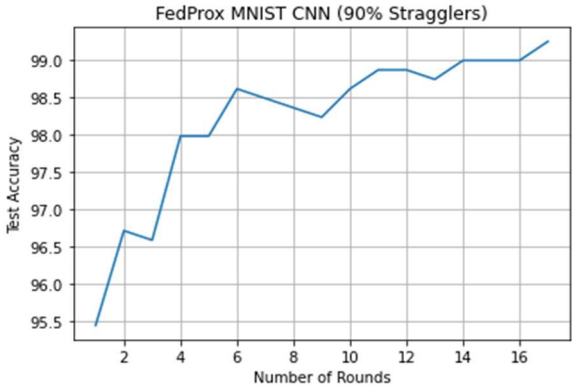


Comparison of FedAvg and FedProx on IID Dataset Partition

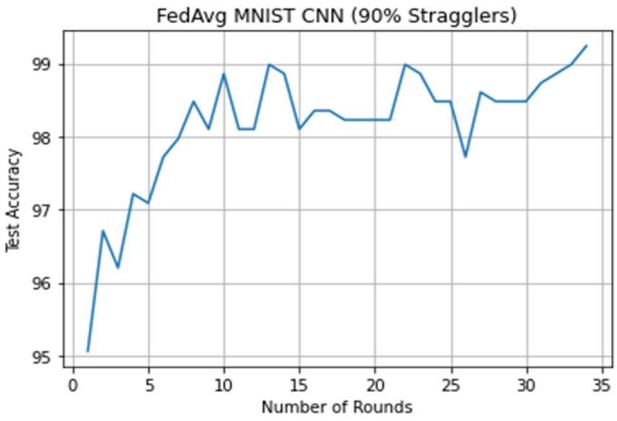
tests to achieve target accuracy on FedAvg and FedProx under a highly heterogeneous(90 %) system.

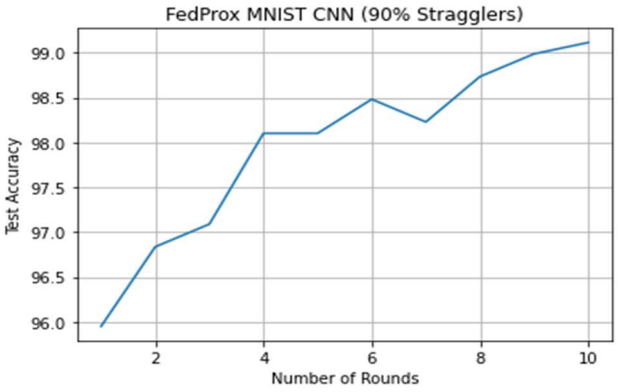
1. lr= 0.01, FedAvg vs FedProx Test_Accuracy on IId



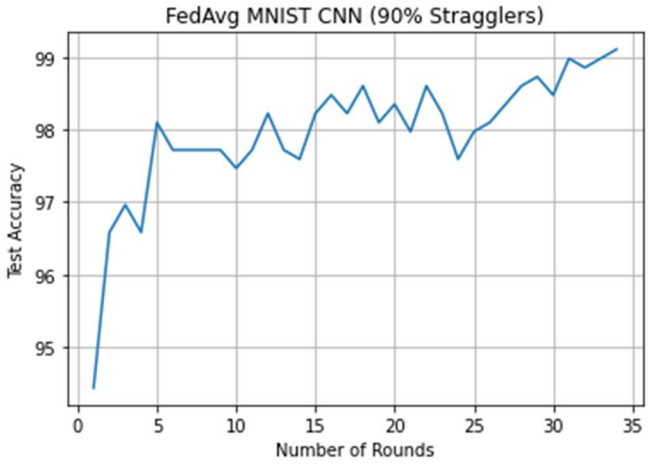


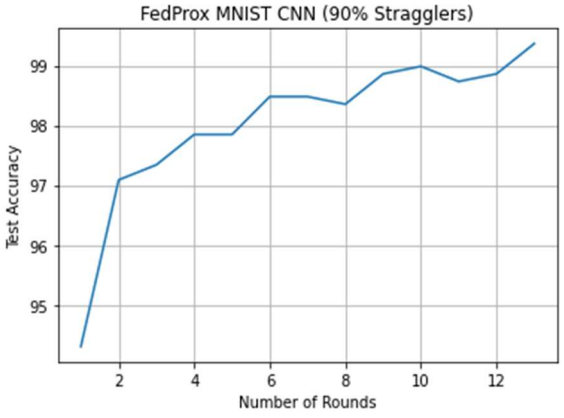
2. lr = 0.03, FedAvg Vs FedProx Test_accuracy on IId





3. lr= 0.05, FedAvg Vs FedProx Test_Accuracy on IId

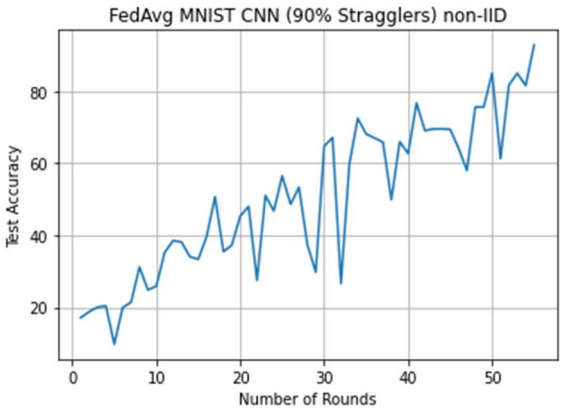


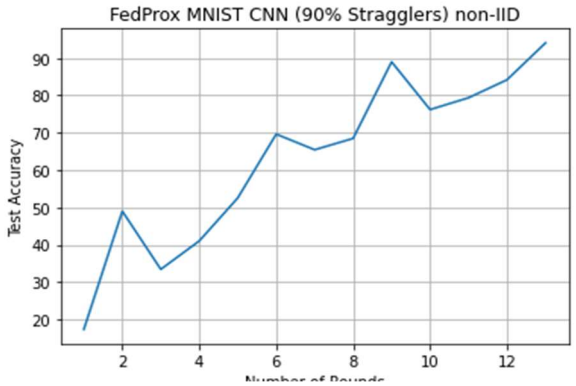


Comparison of FedAvg and FedProx on non-IID Data Set Partition

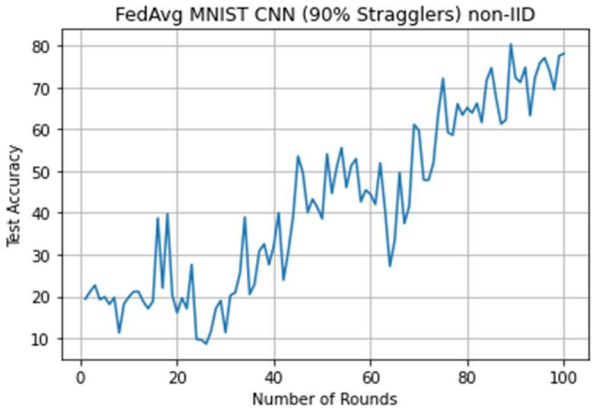
tests to achieve target accuracy on FedAvg and FedProx under a highly heterogeneous system.

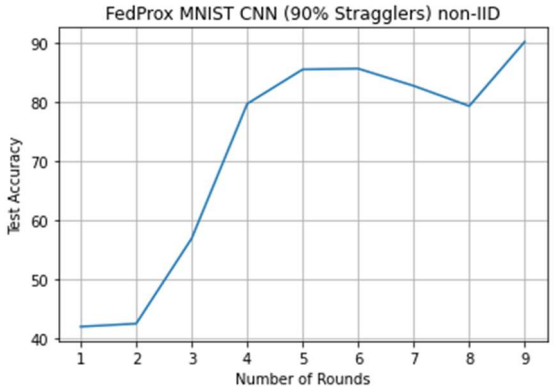
1. lr= 0.01, FedAvg vs FedProx Test_Accuracy on Non-IId



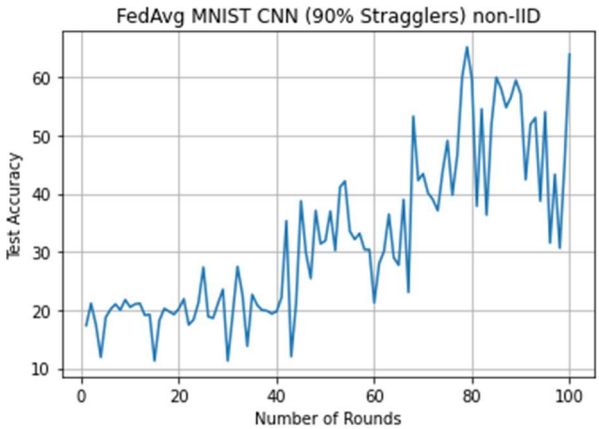


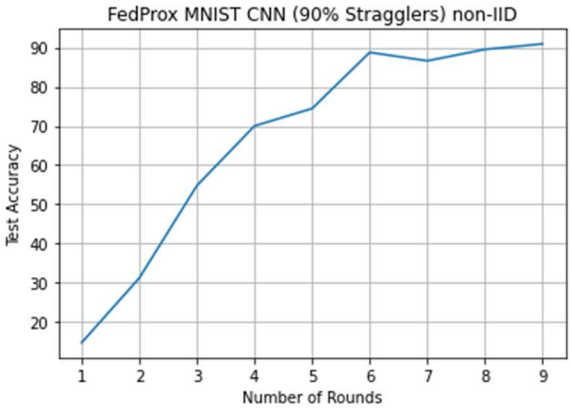
2. lr= 0.03, FedAvg vs FedProx Test_Accuracy on Non-IId





3. lr= 0.05, FedAvg vs FedProx Test_Accuracy on Non-IId





CHAPTER 6

CONCLUSION

As a result, our model can predict outputs on plaintext data as well as the scikit-learn model while maintaining the privacy of all data sources. Although, because of the PHE scheme and more data points, the training time is longer than independent learning, by a factor of 1000 for the breast cancer dataset and 300 for the grad admission dataset. This is dependent on the dataset's size and the number of parties involved. We propose a protocol that ensures that no single party (including the server) can infer anything about the data or its source, protecting the privacy of the patients.

Through the project, we learnt about various Federated Learning Algorithms. When we have IID dataset the model performance is way better in comparison to non-iid dataset, in the real world scenario sometimes due to the various obstacles for example we picked up the 20 mobile user as clients who are connected through internet such that our FL system can do computations and lets say we want to perform keyword correction, in this scenario sometimes it happens out of 20 few number of clients are getting disconnected in between so the data distribution will be affected accordingly that is discussed as heterogeneity.

So we did experimentations around iid as well as non-IID dataset to match with a certain targeted accuracy, iid dataset performance was better in comparison to non-iid.

References

1. Changchang Liu, Supriyo Chakraborty, and Dinesh Verma. Secure model fusion for distributed learning using partial homomorphic encryption. In Policy-Based Autonomic Data Governance, pages 154–179. Springer, 2019.
2. Irene Giacomelli, Somesh Jha, Marc Joye, C David Page, and Kyonghwan Yoon. Privacy-preserving ridge regression with only linearly-homomorphic encryption. In the International Conference on Applied Cryptography and Network Security, pages 243–261. Springer, 2018.
3. Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Scalable and secure logistic regression via homomorphic encryption. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, pages 142–144, 2016.
4. Communication-Efficient Learning of Deep Networks from Decentralized Data
5. Federated optimaization in Federated Networks.