Linköping University | Department of Computer and Information Science

Master's thesis, 30 ECTS | Computer Science and Engineering

2021 | LIU-IDA/LITH-EX-A--21/026--SE

Towards Peer-to-Peer Federated Learning: Algorithms and Comparisons to Centralized Federated Learning

Mot Peer-to-Peer Federerat Lärande: Algoritmer och Jämförelser med Centraliserat Federerat Lärande

Dylan Mäenpää

Supervisor : David Bergström
Examiner : Fredrik Heintz

External supervisor : Simon Almgren, Fredric Landqvist

LINKÖPING
UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

Due to privacy and regulatory reasons, sharing data between institutions can be difficult. Because of this, real-world data are not fully exploited by machine learning (ML). An emerging method is to train ML models with federated learning (FL) which enables clients to collaboratively train ML models without sharing raw training data. We explored peer-to-peer FL by extending a prominent centralized FL algorithm called Fedavg to function in a peer-to-peer setting. We named this extended algorithm FedavgP2P. Deep neural networks at 100 simulated clients were trained to recognize digits using FedavgP2P and the MNIST data set. Scenarios with IID and non-IID client data were studied. We compared FedavgP2P to Fedavg with respect to models' convergence behaviors and communication costs. Additionally, we analyzed the connection between local client computation, the number of neighbors each client communicates with, and how that affects performance. We also attempted to improve the FedavgP2P algorithm with heuristics based on client identities and per-class F1-scores. The findings showed that by using FedavgP2P, the mean model convergence behavior was comparable to a model trained with Fedavg. However, this came with a varying degree of variation in the 100 models' convergence behaviors and much greater communications costs (at least 14.9x more communication with FedavgP2P). By increasing the amount of local computation up to a certain level, communication costs could be saved. When the number of neighbors a client communicated with increased, it led to a lower variation of the models' convergence behaviors. The FedavgP2P heuristics did not show improved performance. In conclusion, the overall findings indicate that peer-to-peer FL is a promising approach.

Acknowledgments

I would like to express my deepeest gratitude to David Bergström and Simon Almgren for the weekly guidance, inspiration and insightful suggestions. I would also like to extend my sincere thanks to Fredrik Heintz and Fredric Landqvist for all the valuable input and helpful advice. Finally, I'd like to thank my family and partner for always believing in me throughout my journey in life.

Contents

List of Figures

List of Tables

1 Introduction

This chapter starts with the motivation for the thesis in Section 1.1. Section 1.2 covers the overall aim of the thesis and the research questions. Delimitations are presented in Section 1.3. Contributions are given in 1.4. Finally, an overview of the overall structure of the thesis is given in Section 1.5.

1.1 Motivation

Recent advances in machine learning (ML) have led to highly performing innovations in numerous fields. For example, in the medical specialty dermatology, an ML model has been used in skin cancer diagnosis and performs at the same level as dermatologists [6]. Furthermore, many of the recent prominent ML applications utilize deep learning [7] which is highly dependent on sufficiently large and diverse data sets to be reliable [24]. However, collecting such types of data sets can be difficult. In several domains, data are owned by many clients and stored at different locations. Due to privacy and regulatory reasons, sharing of data across clients is hindered. Problems with sharing data make it difficult to generate robust ML models that have been exposed to diverse data. Consequently, existing data already collected are not fully capitalized by ML. This is unfortunate since with robust ML models, better efficiency and reduced costs can be achieved in numerous domains, e.g., in healthcare [26, 18].

A method that enables clients to collaboratively train ML models without sharing any raw training data is *federated learning* (FL) [16]. Normally, ML models are trained at one location where the owner of the model can freely observe all the training data. However, in FL, the training of a model is decentralized. The predominant FL strategy utilizes a central orchestrating server that distributes a global model to participating clients. These clients subsequently train the models with their local data. The updated local model parameters are then sent to the central server, where the global model is updated by aggregating and combining the clients' model parameters. In domains where data are sensitive and spread across clients, FL supports the collaboration and usage of data since FL reduces the risk of raw local data being observable by other clients.

The area of FL has in recent years attracted immense interest both from academia and industry. In the industry, some major tech companies have adopted FL in production, and numerous startups are intending to use FL to address regulatory and privacy problems [10]. However, there are numerous challenges with FL such as communication efficiency, systems heterogeneity, non-identically distributed (non-IID) client data, and privacy concerns [13]. For example, non-IID client data (e.g., skewed label distribution) can greatly inhibit the learning process [13].

Centralized FL, where a centralized server orchestrates the learning process is the predominant FL approach [13]. Using centralized FL, clients are required to trust and depend on one central server. This approach carries the risk of disrupting the training process if the server were to fail. Also, in FL scenarios where there are potentially a high number of participating clients, the central server must be able to handle a high amount of communication which can be a limiting factor [13]. To address some issues of centralized FL, peer-to-peer FL could be a viable alternative since it circumvents the central-server dependency. This thesis aims to study the viability of peer-to-peer FL. To do this, we extend a prominent centralized FL algorithm called *Fedavg* [16] to work in a peer-to-peer setting. This extension is inspired by other work that studies decentralized training of models [8, 15, 9].

Furthermore, we will study peer-to-peer FL by training deep neural networks (DNNs). This is motivated by the success of neural networks in many tasks, for example, the aforementioned skin cancer diagnosis ML model that performed at the same level as the dermatologists was a DNN [6]. Thus, DNNs will be studied in conjunction with peer-to-peer FL and will be trained to recognize digits using images from a data set named MNIST [12].

1.2 Aim and research questions

The overarching aim of this thesis is to study the viability of peer-to-peer FL. To do this, we first extend the Fedavg algorithm [16] to work in a peer-to-peer setting. We name this extended algorithm *FedavgP2P*. Comparisons of FedavgP2P will be made to Fedavg through empirical evaluation. Aspects that will be considered are models' convergence behavior and communication costs. These aspects will be examined in scenarios where there are IID and non-IID local client data. Furthermore, this thesis aims to study possible ways to improve FedavgP2P. Given this aim, the following research questions form the foundation of the thesis:

1. How does FedavgP2P compare to Fedavg concerning models' convergence behaviors with IID and non-IID client data?
2. How does FedavgP2P compare to Fedavg concerning communication costs with IID and non-IID client data?
3. Considering FedavgP2P: how does the number of local epochs, and the number of neighbors each client communicates with every round affect the models' convergence behaviors and communication costs?
4. How can FedavgP2P be enhanced such that models' convergence behaviors improve and communication costs decrease?

1.3 Delimitations

There are many different algorithms for centralized FL, in this thesis, we focus on the most prominent: Fedavg [16]. Furthermore, the peer-to-peer FL algorithms we study are our extensions of Fedavg. Regarding communication costs, the number of models sent in the network, and the number of communications rounds will be considered. Other aspects, such as the size of the transmitted data between clients will not be examined. Considering non-IID client data, we will only study skewed distributions of labels at clients. Finally, in this thesis, we assume a good connection between the participating clients in all the experiments.

1.4 Contributions

Our main contributions are: 1) an extension of the centralized FL algorithm, Fedavg, to a peer-to-peer setting; 2) empirical evaluation of the extension and comparisons to its centralized counterpart. More specifically, we introduce the algorithm FedavgP2P. We conduct several experiments on FedavgP2P, demonstrating the performance and differences to its centralized counterpart with both IID and non-IID client data. The results indicate that FedavgP2P can be a viable approach to train DNNs across multiple clients.

1.5 Structure

The thesis consists of six chapters. The first chapter is the present introductory chapter. Chapter 2 presents the theoretical background which covers machine learning, federated learning, evaluation metrics, and related work. Chapter 3 presents the FedavgP2P algorithm and FedavgP2P enhancements through heuristics. Chapter 4 introduces the method used to answer the research questions. This includes information about software and hardware, data, neural network architecture, experiments, and evaluation methods. Chapter 5 covers the results from the experiments. Chapter 6 discusses the results, the method, and the work in a wider context. Finally, Chapter 7 concludes the thesis by giving a summary of the findings. Additionally, Chapter 7 presents suggestions of future work.

2 Theory

In this chapter, we first introduce machine learning in Section 2.1. Section 2.2 presents federated learning along with the Fedavg algorithm and challenges with federated learning. Model evaluation metrics are given in Section 2.3. Finally, related work is presented in Section 2.4.

2.1 Machine learning

Machine learning (ML) is a branch of artificial intelligence where computers learn patterns in data and draw conclusions. ML can be categorized into three subcategories: 1) *supervised learning*, 2) *unsupervised learning*, and 3) *reinforcement learning*. In this thesis, we only consider supervised learning.

In supervised learning, a model is trained using labeled data which is used to reduce an ML model error. Let us consider an example in the task of classification, where a model is trained to identify cats and dogs. Here, the input to the model is labeled images of cats and dogs. Using supervised learning algorithms, the model parameters are tuned to better recognize cats and dogs by utilizing the labeled training data.

In this thesis, we consider ML algorithms that are applicable to a finite-sum objective [16]:

$$\min_{w \in R^d} f(w), \qquad \text{where} \qquad f(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w). \tag{2.1}$$

For a supervised learning problem, the function $f_i(w)$ is typically treated as a loss function, i.e. $f_i(w) = \ell(x_i, y_i; w)$, where $\ell(x_i, y_i; w)$ is the loss of the prediction on an input-output pair training example (x_i, y_i) with model parameters w. Here, x_i contains the example features and y_i contains the label. If we relate this to the previous example with the classification of dogs and cats, x_i represents the pixels of the image and y_i the label cat or dog. The function objective in a supervised learning problem can be described as finding w which minimizes the average loss over all training input samples n. The process of finding an optimal w is typically referred to as *training*.

There are many different types of ML models and associated training algorithms. Examples of models are: *Support vector machines*, and *neural networks*. In this thesis, we consider neural networks (NNs). In NNs, neurons are connected, where the output of a neuron is used as input to other neurons. Each neuron can be associated with a weight that can be tuned to fit training data. Moreover, the term deep neural networks (DNNs) refers to a neural network with multiple layers of neurons.

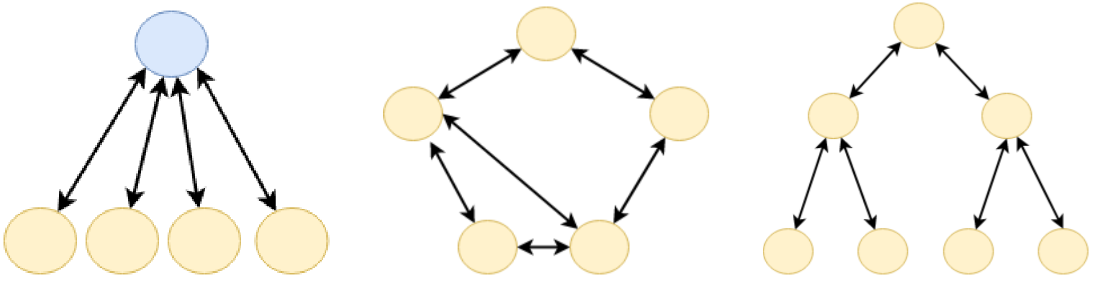
We can use the finite sum-objective presented in Equation 2.1 to formalize the objective of a neural network. Let w be the weights of a NN. An approach to find w is by using optimizers. A typical example of an optimizer is Stochastic Gradient Descent (SGD) which computes the loss gradients $\nabla f_i(w)$ with backpropagation [7]. Moreover, there is a myriad of ways to build neural networks and there are many settings that the practitioner can choose. These include the choice of the loss function, activation function, number of neurons, layer types, and number of layers.

2.2 Federated learning

ML models are normally trained in a centralized manner where the data are stored at one client and the owner of the model can freely observe the data. Collecting diverse and rich data sets can in many cases be difficult due to the data being sensitive. This makes it hard to train robust deep learning models which need sufficiently large and diverse data sets [24]. As an answer to this problem, McMahan et al. [16] introduce FL which decentralizes the training of ML models. In FL, an arbitrary amount of clients can participate in the training of an ML model. Each client trains a model with their local data and shares the model updates (e.g., model parameters) with other clients. The model updates can then be utilized and aggregated at other locations. A key characteristic of FL is that local raw data are never shared among clients, only the model updates are shared. This minimizes the risk of raw data being exposed. In a federated setting, there are often numerous challenges such as unbalanced non-IID client data, a high number of participating clients, and high communication costs [16]. See Section 2.2.5 for more information about challenges in federated settings.

2.2.1 Definition

Similarly to Yang et al. [25] we define FL as K clients owning their respective data $\{P_0, P_1, ..., P_K\}$. The clients collaboratively train a machine learning model using other clients' data without any client i exposing its raw data P_i to others.



(a) Centralized topology (b) Peer-to-peer topology (c) Hierarchical topology

Figure 2.1: Three different network topologies. The yellow circles represent clients, and the blue circle represents a central orchestrating server.

Moreover, following McMahan et al. [16] we formalize FL by re-writing the objective in Equation 2.1. The objective function $f(w)$ in Equation 2.1 is modified to $f(w)$ in 2.2.

$$f(w) := \frac{1}{n}\sum_{k=1}^{K} \frac{n_k}{n} F_k(w), \qquad \text{where} \qquad F(w) := \frac{1}{n_k}\sum_{i \in P_k} f_i(w). \tag{2.2}$$

Here K clients are assumed, each client k owning their data P_k. Each client computes $F_k(w)$, which is the average loss on client k. The number of training samples on each client is denoted by n_k, i.e. $n_k = |P_k|$. The number of total samples on all K clients is denoted by n.

2.2.2 Centralized federated learning

McMahan et al. proposed an FL algorithm called Fedavg [16] that depends on a central orchestrating server to which all participating clients are connected. The network topology is thus centralized. A constellation where a central server orchestrates the FL process is referred to as *centralized FL* in this thesis. An illustration of the centralized network topology is presented in Figure 2.1. Since the introduction of Fedavg, other centralized FL algorithms have been introduced to tackle the various amount of scenarios. One of those is Fedprox [14], which can work better with non-IID client data according to the authors. A further description of Fedprox is presented in Section 2.4.2.

2.2.3 Federated averaging

In Fedavg [16, 11] a global model is first initialized. Then, every round t, the central orchestrating server sends out the current global model w_t to a selected fraction C of all the participating clients K. These selected clients are expressed as the set S_t. Each client k then trains the model on its local data P_k, resulting in a model w_{t+1}^k, and sends its updated local model parameters to the central server. Then, the central server aggregates and averages the received model parameters to generate a new global model $w_{t+1} = \sum_{k \in S_t} \frac{n_k}{n_t} w_{t+1}^k$ where n_t is the total number of all samples from the selected clients, and n_k is the number of samples at client k. When training is complete, the central server sends the global model to all the clients in the network. Furthermore, Fedavg has four hyperparameters, the fraction of clients C to selected for each round, B the local minibatch size, E the number of times each client train over the local data set each round (i.e. epochs), and the learning rate η. Fedavg is presented in Algorithm 1.

Algorithm 1: Federated averaging (Fedavg) [16]

```
Server executes:
    initialize $w_0$
    foreach round $t = 1, 2, ...$ do
        $m = max\{C \cdot K, 1\}$
        $S_t$ = (random set of $m$ clients)
        foreach client $k \in S_t$ in parallel do
            $w_{t+1}^k$ = ClientUpdate($k$, $w_t$)
        $w_{t+1} = \sum_{k \in S_t} \frac{n_k}{n_t} w_{t+1}^k$

Function ClientUpdate($k, w$): // Run on client $k$
    $\beta$ = (split $P_k$ into batches of size $B$)
    foreach local epoch $i$ from 1 to $E$ do
        foreach batch $b \in \beta$ do
            $w = w - \eta \nabla \ell(w; b)$
    return w to server
```

2.2.4 Decentralized federated learning

The central server in centralized FL can suffer from communication and computational bottlenecks due to a high amount of connected clients. Furthermore, if the centralized server fails, the training process can be disrupted.

In decentralized FL, a central orchestrating server is not needed, thus circumventing the aforementioned issues. In this instance, clients only communicate with their neighbors. An illustration of two decentralized network topologies, peer-to-peer and hierarchical, is presented in Figure 2.1.

2.2.5 Problems in federated learning

Li et al. [13] identify four core challenges with FL. Firstly, *expensive communication* can occur between the clients due to a potentially high number of participating clients in a federated network. Two important aspects of the cost are the number of communication rounds and the size of the transmitted data (e.g., model parameters). Secondly, *systems heterogeneity* among participating clients regarding differing hardware and network capabilities introduces problems such as stragglers which can slow down the training process. Thirdly, *statistical heterogeneity* in data is a natural outcome in many domains. For instance, two clients collect data from different populations. The non-IID client data can impact the performance of models trained with FL negatively compared to a model trained with a traditional approach. Fourthly, there are *privacy concerns*. ML models have a risk of leaking sensitive information. For example, previous work by Carlini et al. [3] show that sensitive text patterns can be extracted from a recurrent neural network.

2.2.6 Statistical heterogeneity

In real-world scenarios, federated data sets could contain heterogeneous client data (non-IID data). The non-IID client data at be expressed in many forms [10]:

- Skewed distribution of labels at clients. For example, one client has only collected a subset of all possible labels.
- Skewed distribution of features, e.g., some clients have more features than others.
- Skewed quantity. Clients can hold a different amount of samples.
- Varying relationships between features and labels. For example, one client labels some features as a horse, whereas another client labels the same features as an animal.

2.3 Model evaluation metrics

There are many evaluation metrics to choose from when evaluating ML models. The various metrics measure different aspects of an ML model, thus an ML model can perform well on one metric and worse on other metrics. In some cases, it is important to evaluate ML models with varying metrics. Following the definitions from Sokolova and Lapalme [22], the metrics *accuracy*, *recall*, *precision*, and *F-score* are introduced in this section for both binary- and multi-class classification.

Before introducing the metrics, the concepts of true positives (TP), false negatives (FN), true negatives (TN), false positives (FP), and F1-score are defined. In the case of the binary classification problem, the outcome can either be positive or negative. If the model predicts the positive outcome correctly, it is a TP. Conversely, if the model predicts the positive outcome incorrectly, it is an FN. Similarly, a TN occurs when the model predicts a negative outcome correctly, and FP occurs when the model predicts a negative outcome incorrectly. Furthermore, a confusion matrix showing the concepts of a TP, FN, TN, and FP in a binary classification problem is shown in Figure 2.2. In a multi-class classification problem, the concept of TP, FN, TN, FP in binary classification problems is applied for every class and the results are then combined.

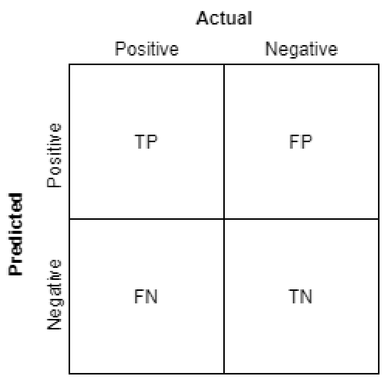


Figure 2.2: A confusion matrix for the binary classification problem.

Accuracy is defined as the fraction of predictions the model predicted correctly. Equations are presented in 2.3 and 2.4 for the binary and multi-class classification cases respectively. K is the total amount of classes.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

$$Average\ Accuracy = \frac{\sum_{i=1}^{K} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}}{K} \tag{2.4}$$

Precision is defined as the proportion of correctly predicted positives by the model. This is defined for the binary classification case in Equation 2.5.

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

Recall is defined as the fraction of all positive samples that were correctly classified as positive. This is defined in Equation 2.6 for the binary classification case.

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

F-score combines the two measures recall and precision. A formal definition for the binary classification problem is given in Equation 2.7. The β variable weighs the relative importance of precision and recall. If $\beta = 1$ (F1-score), the F-score is balanced and equal weight is put on precision and recall. For the multi-class classification problem, the F-score can be evaluated in two ways. Firstly, macro-averaging (denoted with M) presented in Equations 2.8, 2.9 and 2.10, computes the precision and recall for each class independently and then computes the averages. Hence all classes are treated equally. Secondly, micro-averaging (denoted with μ) presented in Equations 2.11, 2.12 and 2.13, aggregates all contributions from all classes first and then computes the averages. This could be preferable if there is a class imbalance in the samples.

$$F\text{-}score = \frac{(\beta^2 + 1) \cdot TP}{(\beta^2 + 1) \cdot TP + \beta^2 \cdot FN + FP} \tag{2.7}$$

$$Precision_M = \frac{\sum_{i=1}^{K} \frac{TP_i}{TP_i + FP_i}}{K} \tag{2.8}$$

$$Recall_M = \frac{\sum_{i=1}^{K} \frac{TP_i}{TP_i + FN_i}}{K} \tag{2.9}$$

$$F\text{-}score_M = \frac{(\beta^2 + 1) \cdot Precision_M \cdot Recall_M}{\beta^2 \cdot Precision_M + Recall_M} \tag{2.10}$$

$$Precision_\mu = \frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{K} (TP_i + FP_i)} \tag{2.11}$$

$$Recall_\mu = \frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{K} (TP_i + FN_i)} \tag{2.12}$$

$$F\text{-}score_\mu = \frac{(\beta^2 + 1) \cdot Precision_\mu \cdot Recall_\mu}{\beta^2 \cdot Precision_\mu + Recall_\mu} \tag{2.13}$$

2.4 Related work

2.4.1 Decentralized topology

He et al. [8] study training of linear models in a fully decentralized environment. A decentralized algorithm is proposed, and convergence guarantees for the algorithm are theoretically proven for convex objectives. In the algorithm, clients compute and update a local solution to a subproblem, and share the solution with neighboring clients. Moreover, the authors demonstrate that the algorithm offers adaptivity to dynamic networks and system heterogeneous scenarios by tuning a client-specific local parameter based on the client resources. Furthermore, the algorithm is studied in experiments with 16 clients in 5 different graph topologies: 2-connected cycle, 3-connected cycle, 2D grid, and complete graph. In all experiments, the algorithm converges monotonically. In this study, the experiments are limited since they only consist of 16 clients in the network and few variations of graph topologies. Additionally, the study only considers convex problems and linear models, thus the implications are not necessarily generalizable to non-convex problems, e.g., deep learning.

Under poor communication networks, decentralized learning is faster than its centralized counterpart [15]. Lian et al. [15] theoretically and empirically show that decentralized SGD is faster than centralized SGD in low-bandwidth networks. This work does not take typical federated settings into account, such as non-IID client data.

Gossip learning [19] is a related area to FL that trains ML models without a central orchestrating server. Instead, clients in the network communicate and share model parameters with peers through a peer-sampling service, for example by taking random walks [23]. Hegedűs et al. [9] compare gossip learning to centralized FL. Linear models are trained in several scenarios where a fixed random network was generated. Each client in the network had 20 neighbors with whom they communicated to. A comparison was made to centralized FL in various scenarios. In some scenarios where compression techniques were used for the shared model parameters, gossip learning showed competitive results to FL with respect to communication costs. However, without compression techniques, FL outperforms gossip learning. Hegedűs et al. [9] only consider linear models, thus the implications are not generalizable to non-convex problems such as deep learning.

Biggs et al. [1] present an FL approach that builds hierarchical clusters, with the aim to build more personalized models for clients. The clusters are chosen by the similarity of clients' local model parameters. The clustering method starts by vectorizing all local model parameters which represent singleton clusters. Then, for each step in the clustering method the two most similar clusters, measured by pairwise distance, are merged. Each cluster is thus more personalized for the clients residing in each cluster. The authors show that in a variety of IID and non-IID scenarios, their approach converges more quickly than Fedavg when training a DNN, using fewer communications rounds. The drawback of the approach is that the clustering method assumes that all clients are connected to one central coordinating server at the beginning, which is not always possible in all scenarios. In addition, due to the clusters merging local models with similar model parameters, the data distributions at the clients in the cluster are alike. Thus, the independently trained cluster models have not been exposed to the global data distribution, which yields less general robust models.

2.4.2 Heterogeneity

Two key challenges in federated learning are systems heterogeneity among clients and non-IID client data [13]. To face these challenges, Li et al. [14] introduced an algorithm, Fedprox, that can be viewed as a re-parametrization of Fedavg. In Fedavg, the local number of epochs at all clients is fixed. However, Li et al. find that the best number of local epochs is likely to change during each communication round. Further, the best choice of local epochs is dependent on the local client data set and system resources. Choosing a dynamic number of local epochs for each client is therefore a key property of the Fedprox algorithm. Performing too many local epochs can cause Fedavg to diverge [16], thus a proximal term is proposed in Fedprox to reduce the impact of local updates by restricting them to be closer to the global model.

3 FedavgP2P: A Peer-to-Peer Federated Learning Algorithm

In this chapter we introduce FedavgP2P in Section 3.1. Additionally, we present three FedavgP2P heuristics in Section 3.2.

3.1 Algorithm

In Fedavg (see Section 2.2.3), there is a need for a central orchestrating server. Inspired by other work with decentralized training algorithms [8, 15, 9], we extend Fedavg to work in a peer-to-peer setting, thus eliminating the need for a central server. The extended algorithm is further referred to as *FedavgP2P*. In FedavgP2P, each client have their own model and communicate directly to other clients. Before training, all client models are initialized with the same weights w_0. Every round t, each client c trains the model on its local data P_c, resulting in a model w_t^c. Then, each client aggregates and averages updates from a set of random neighbors S_t where $|S_t|$ is calculated by $C \cdot A$, where C is the fraction of neighbors (e.g., 0.1) and A is the total number of neighbors in the network for that client. Next, the local model is updated $w_{t+1}^c = \frac{n_c \cdot w_t^k}{n_t} + \sum_{k \in S_t} \frac{n_k}{n_t} w_t^k$ where n_t is the total number of samples from client c and from the clients in S_t, and n_k is the number of samples at neighbor k. Similar to Fedavg, FedavgP2P has four hyperparameters, the fraction of neighbors C which each client receives updates from, B the local minibatch size, E the number times each client train over the local data set each round (i.e. epochs), and the learning rate η. FedavgP2P can be found in Algorithm 2.

Algorithm 2: Federated averaging peer-to-peer (FedavgP2P)

```
1  **Client** $c$ **executes:**
2      initialize $w_0^c$
3      **foreach** *round* $t = 1, 2, ...$ **do**
4          $\beta$ = (split $P_c$ into batches of size $B$)
5          **foreach** local epoch $i$ from 1 to $E$ **do**
6              **foreach** batch $b \in \beta$ **do**
7                  $w_t^c = w_t^c - \eta \nabla \ell(w_t^c; b)$
8          $m = max\{\lceil C \cdot A \rceil, 1\}$
9          $S_t$ = (random set of $m$ neighbors)
10         **foreach** client $k \in S_t$ **in parallel do**
11             $w_t^k$ = GetWeights(k) // **Get weights from client k**
12         $w_{t+1}^c = \frac{n_c \cdot w_t^c}{n_t} + \sum_{k \in S_t} \frac{n_k}{n_t} w_t^k$
```

3.2 Heuristics

In the original FedavgP2P, the neighbors a client communicates with are chosen randomly. To attempt to improve the performance of FedavgP2P, we present three different heuristics for choosing neighbors to communicate with. Descriptions of these heuristics follow in the subsequent sections.

3.2.1 Heuristic 1

In the first heuristic, we introduce *client identities*. Each client has its own identity, and each client stores the identity of the 10 latest clients it has communicated with. After each communication round, the information about the 10 latest clients is broadcasted out to the network. Then, every client chooses neighbors to communicate with by choosing those who have communicated most differently. We formalize the heuristic by letting the set Y_c denote the 10 latest neighbors client c has communicated with. Then, client c chooses the neighbors to communicate with based on the neighbors with most different sets, i.e. for every neighbor k we compute $|Y_c \backslash Y_k|$. Neighbors that communicated most differently have the highest values considering $|Y_c \backslash Y_k|$. The extended FedavgP2P algorithm with heuristic 1, which we further refer to as *FedavgP2P 10 latest*, can be observed in Algorithm 3. The adapted or added rows are 9, 12, and 14.

Algorithm 3: FedavgP2P 10 latest

```
1  Client $c$ executes:
2      initialize $w_0^c$
3      foreach round $t = 1, 2, ...$ do
4          $\beta$ = (split $P_c$ into batches of size $B$)
5          foreach local epoch $i$ from 1 to $E$ do
6              foreach batch $b \in \beta$ do
7                  $w_t^c = w_t^c - \eta \nabla \ell(w_t^c; b)$
8          $m = max\{\lceil C \cdot A \rceil, 1\}$
9          $S_t$ = DifferentNeighbors(c, m) // Get $m$ most different neighbors
10         foreach client $k \in S_t$ in parallel do
11             $w_t^k$ = GetWeights(k) // Get weights from client k
12             $latest\_clients$ = Save10LatestClients(c, k)
13         $w_{t+1}^c = \frac{n_c \cdot w_t^c}{n_t} + \sum_{k \in S_t} \frac{n_k}{n_t} w_t^k$
14         Broadcast($c$, $latest\_clients$)
```

3.2.2 Heuristic 2 and 3

In addition to the client identities in the first heuristic, the second and third heuristic utilizes the models' performances. The intuition behind these heuristics is to make clients communicate with neighbors that have models that perform better or are dissimilar to their own model. Here, after each communication round, each client computes their model per-class F1-score on a test set. Furthermore, the per-class F1-scores are broadcasted out to the network. For a client to choose which neighbors to communicate with, a dissimilarity or similarity score was computed for every neighbor based on the per-class F1-scores. Here, two choices of computing scores were adopted which corresponds to heuristic 2 and 3.

Heuristic 2) Let F_c^i denote the F1-score on class i at client c. For each client c a dissimilarity score for each neighbor k was computed by:

$$neighbor\ dissimilarity\ score = \sum_{i \in D} (F_k^i - F_c^i) \tag{3.1}$$

where D denotes the set of classes. Client c will then choose the neighbors to communicate with that have the highest total scores.

Heuristic 3) We treat the F1-scores for client c as 10-dimensional vectors F_c. Then, for each client c the *cosine similarity* was computed for every neighbor k:

$$cos(\theta) = \frac{F_c \cdot F_k}{||F_c||\ ||F_k||} \tag{3.2}$$

The lower the value is, the more dissimilar the neighbor is considering the cosine similarity. Thus, client c chooses to communicate with the neighbors with the lowest cosine similarity scores.

The adapted FedavgP2P algorithm that describes heuristic 2 and 3, which we further refer to as *FedavgP2P F1-score arithmetic* and *FedavgP2P F1-score cosine*, can be observed in Algorithm

4. The adapted or added rows are 9, 13, and 14. Note that the function on row 9 decides which similarity score to use.

Algorithm 4: FedavgP2P F1-score

```
1  **Client** $c$ **executes:**
2      initialize $w_0^c$
3      **foreach** $round\ t = 1, 2, ...$ **do**
4          f $\beta$ = (split $P_c$ into batches of size $B$)
5          **foreach** local epoch $i$ from 1 to $E$ **do**
6              **foreach** batch $b \in \beta$ **do**
7                  $w_t^c = w_t^c - \eta \nabla \ell(w_t^c; b)$
8          $m = max\{\lceil C \cdot A \rceil, 1\}$
9          $S_t$ = NeighborsFscore(c, m) **// Get** $m$ **neighbors based on F1-scores**
10         **foreach** client $k \in S_t$ **in parallel do**
11             $w_t^k$ = GetWeights(k) **// Get weights from client k**
12         $w_{t+1}^c = \frac{n_c \cdot w_t^c}{n_t} + \sum_{k \in S_t} \frac{n_k}{n_t} w_t^k$
13         $class\_fscores$ = TestModel($w_{t+1}^k$)
14         Broadcast($c$, $class\_fscores$)
```

4 Method

To fulfill the thesis aim and to answer the research questions, we conducted numerous experiments. This chapter begins with a description of the hardware and software used in Section 4.1. In Section 4.2 the data are presented. Section 4.3 covers the neural network architecture and design. Experiments are explained in Section 4.4. Finally, evaluation methods are presented in Section 4.5.

4.1 Hardware and software

We used a system running the operating system Ubuntu[1] with version 18.04.4 LTS. The system CPU was Intel Core i7-7700 CPU 3.60GHz and GPU Geforce RTX 2080 with 8GB GDDR6 usable memory. All experiments in this thesis were computed on this machine. The GPU was utilized for training and testing the neural networks. Furthermore, scripts were written in Python version 3.8.5 for running the various experiments. Pytorch [20] version 1.7.1 was the framework used for building and training the neural networks. Moreover, CUDA version on the system was 10.2. Python and Pytorch were used due to the vast amount of available libraries and resources for machine learning projects. Before every experiment, we seeded the Python and Pytorch random number generators with the same value.

4.2 Data

The MNIST data set [12] was used which contains images of handwritten digits. MNIST consists of a training set with 60,000 examples of digits written by approximately 250 writers. The writers were high school students or employees from the Census Bureau in the U.S. [12]. Furthermore, MNIST consists of a test set with 10,000 examples. Both the training and the test set were used in this thesis. The images are 28x28 pixel grayscale images. Samples of the digits can be seen in Figure 4.1. The distribution of classes in the training and test set can be observed in Figure 4.2. This data set has previously been used in FL research, e.g., by McMahan et al. [16] which motivated us to use this data set.

[1]https://ubuntu.com/

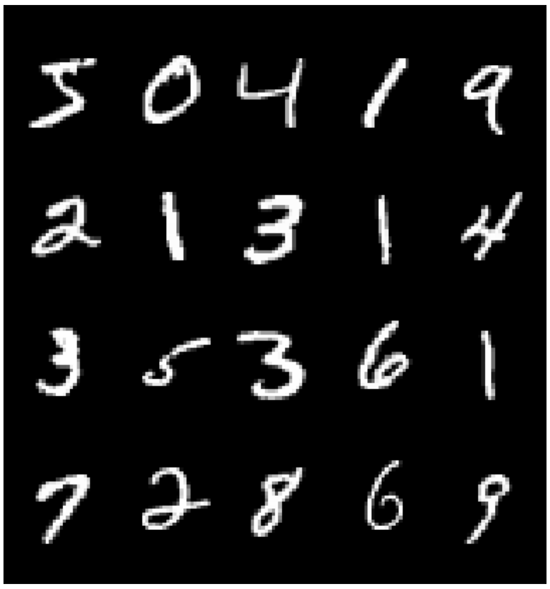


Figure 4.1: Examples of handwritten digits from the MNIST data set [12].

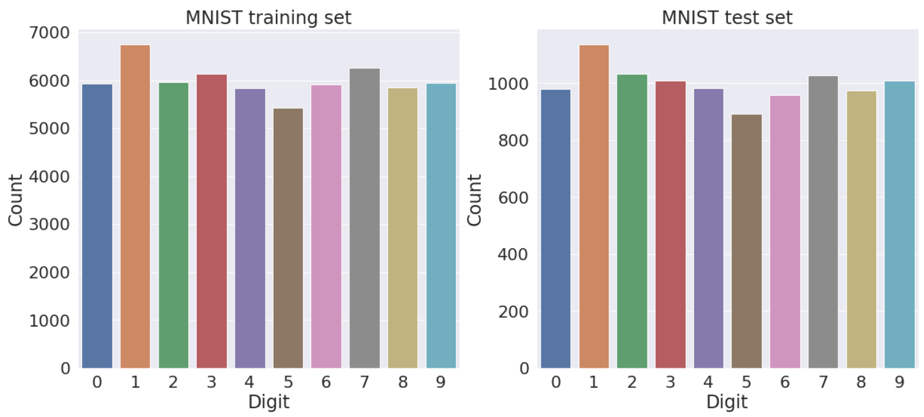


Figure 4.2: The class distribution in the training and test set.

To study FL in situations when there are IID and non-IID client data, we partitioned the MNIST training set over 100 clients in an IID and non-IID manner. The particular way of partitioning data over 100 clients followed from McMahan et al. [16]. In the IID case, each client was given 600 random examples. In the non-IID case, all training samples of digits were divided into 200 equally large sets with only one type of digit. One such set consisted of approximately 300 samples. Each client then received two sets, thus each client held approximately 600 samples of two types of digits. The non-IID partitioning gave a skewed distribution of labels at clients, as described in Section 2.2.6. An example of IID and non-IID data are presented in Figure 4.3.

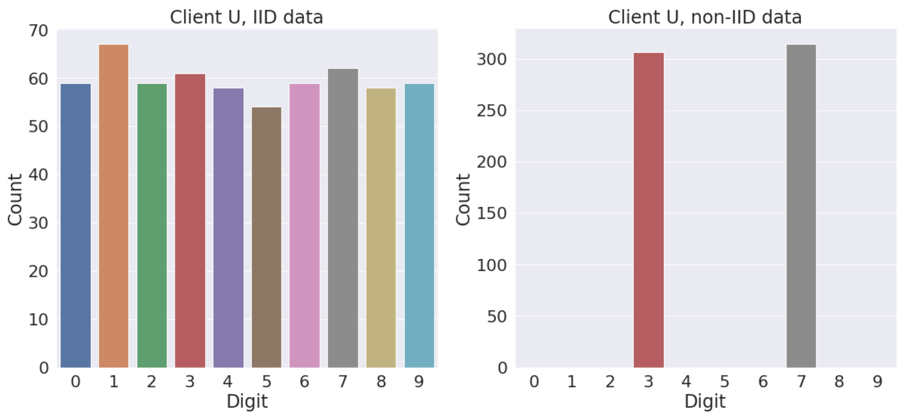


Figure 4.3: An example of client IID and non-IID data.

4.3 Neural networks architecture

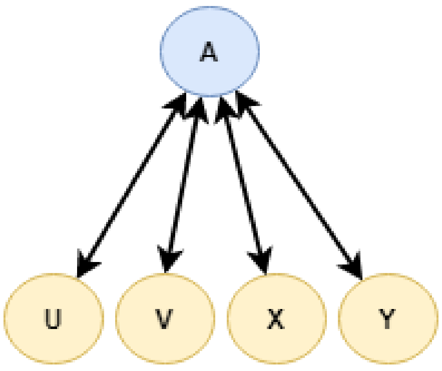
Following McMahan et al. [16], we employed a simple deep neural network with 2-hidden layers containing 100 units each. This neural network is further referred to as *2NN*. Each hidden layer contains 100 units with Rectified Linear Units (ReLu) as their activation function. The MNIST images are 28x28, thus the input layer contains 28x28 = 768 units. One-hot encoding is used for the output classes, meaning there are 10 units in the last layer, one for each digit. The last layer uses the softmax function as the activation function. Furthermore, the cross-entropy loss function is used. The resulting number of parameters in 2NN is 199,120. While this model is not state-of-the-art, 2NN is sufficient for the need to explore how FedavgP2P and Fedavg perform when training DNNs. Furthermore, since they only contain a low amount of parameters compared to more complex architectures, many more experiments can be conducted in less time.

4.4 Experiments

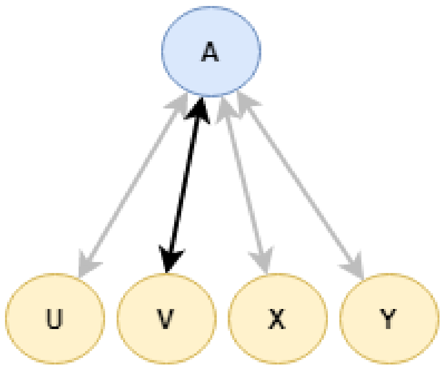
To compare FedavgP2P with Fedavg, we reproduced some of the Fedavg 2NN experiments by McMahan et al. [16]. Further description of those experiments follows in Section 4.4.1. The motivation behind reproducing them was to compute all experiments under the same conditions, thus making the comparisons more reliable. Furthermore, the FedavgP2P experiments are described in Section 4.4.2.

4.4.1 Centralized federated learning

We reproduced a subset of the experiments conducted by McMahan et al. [16] with the 2NN architecture and MNIST data. The algorithm we used was Fedavg which was described in Section 2.2.3. In these experiments, there were 100 clients connected to a central server. As demonstrated by McMahan et al., the most promising results with the 2NN architecture and MNIST data were obtained with batch size $B = 10$, thus we chose to only run the experiments with $B = 10$. We distributed the data over the 100 clients in an IID and non-IID manner as described in Section 4.2. Furthermore, in the experiments with IID client data, we computed a total of 200 communication rounds, and for the non-IID client data, we computed a total of 1,000 communication rounds. One communication round here is defined by two steps: 1) clients selected for the round receives the global model from the central server, 2) selected clients send the updated local model parameters back to the central server. After each communication round, the central aggregated averaged model was evaluated on the MNIST test



(a) All clients are initially connected to the central server A.



(b) An example of a communication round. Central server A sends global model parameters to a fraction of clients, in this case only to client V. V trains the global model on local data and then sends updated local model parameters back to A.

Figure 4.4: A general overview of the Fedavg experiments. The yellow circles represent clients, and blue circles represent a central server.

data. Further description of the model evaluation process follows in Section 4.5.1. Learning rate was set to $\eta = 0.1$. Moreover, we studied how the fraction of clients (C) and the number of local epochs (E) impact model convergence behavior and communication costs, thus we ran experiments with varying C and E. The values of C were set to 0.01, 0.20, 0.50 and 1.00. These fractions mean that the central server communicated with 1, 10, 20, 50, and 100 clients each round. The number of local epochs was set to 1, 5, 10, 20. In total, we ran 40 experiments with Fedavg. After the completion of an experiment, the communication costs were calculated as described in Section 4.5.2. A visual of how centralized FL experiments were conducted is shown in Figure 4.4.

4.4.2 Peer-to-peer federated learning

We conducted experiments with the FedavgP2P algorithm and the additional heuristics (see Chapter 3). To compare FedavgP2P to the Fedavg experiments we used the same amount of clients in the network. A complete graph was constructed, that is, all 100 clients were able to communicate with every other client in the network. A complete graph might not be realistic or even the best choice in some domains, e.g., two clients have a poor connection. However, it is interesting to study how peer-to-peer FL performs in simulated best-case-scenarios networks, such as in a fully connected graph where we assume a good connection between all connected clients. In these experiments, all clients run the same number of epochs before sharing model parameters with others every communication round. The sharing of parameters then happens at the same time for all clients. The following subsections describe how the experiments were conducted with FedavgP2P and the heuristics.

FedavgP2P

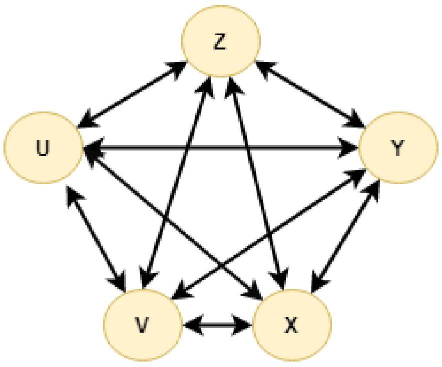
We distributed the data over the 100 clients in an IID and non-IID manner as described in Section 4.2. In the experiments with IID client data, we computed a total of 200 communication rounds with each client, and for the non-IID client data, we computed a total of 1000 communication rounds. After every 10th communication round, we evaluated all client models on the test data. In the FedavgP2P case, we define a communication round as finished when a client has received updates from the neighbors (note that this differs from the centralized

FL case). Further description of the model evaluation process follows in Section 4.5.1. Similar to the centralized FL experiments, we studied how the fraction of neighbors (C) and the number of local epochs (E) impacted model convergence behavior and communication costs, thus we ran experiments with varying C and E. The values of C were set to 0.01, 0.02, 0.05, 0.10, 0.20, 0.50 and 1.00. This means that every client communicated with 1, 2, 5, 10, 20, 50, and 99 neighbors each round. Slightly more variations of C were studied here compared to Fedavg. Peer-to-peer FL is fundamentally different, and the implication of C is not the same in the two algorithms. Each client in peer-to-peer FL can be perceived as a central server. This motivated us to study a larger variety of C in the FedavgP2P experiments. The number of local epochs was set to 1, 5, 10, 20. Moreover, the learning rate was set to the same value as in the Fedavg experiments, i.e. $\eta = 0.1$. In total, we ran 56 experiments with peer-to-peer FL. After the completion of an experiment, the communication costs were calculated as described in Section 4.5.2. A general overview of how the FedavgP2P experiments were conducted is shown in Figure 4.5.

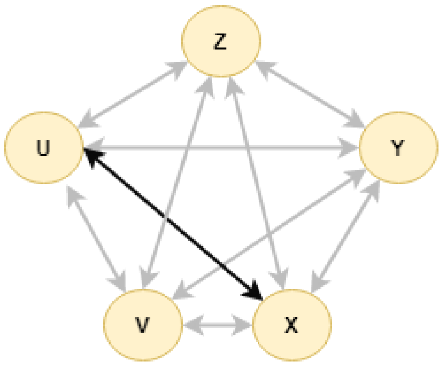
Heuristics

To evaluate the performance of the three heuristics *10 latest*, *F1-score*, and *F1-score cosine*, we adopted the FedavgP2P experiments with non-IID data presented in the section above. However, we only considered the number of local epochs (E) to 5. The IID experiments and other values of E were not considered due to the time constraints of the thesis. We varied the number of fraction of clients (C) to 0.01, 0.02, 0.05, 0.10, 0.20, and 0.50. These fractions mean that each client communicated with 1, 2, 5, 10, 20, and 50 neighbors each round. We did not consider $C = 1.00$ since those experiments are equivalent to FedavgP2P. After every 10th communication round, we evaluated all client models on the original test data. Seven experiments were conducted per heuristic, which totals 21 experiments. After the completion of an experiment, the communication costs were calculated as described in Section 4.5.2.

Regarding the experiments with heuristics *F1-score arithmetic* and *F1-score cosine*, each client computed their model per-class F1-scores on a mini test set containing 1,000 samples after each communication round. This test set was obtained by randomizing 1,000 samples from the original MNIST test set. In practice, we could have used the whole test set, but to reduce computation time for each experiment we only used 1,000 samples. A figure illustrating the smaller test set is found in Figure 4.6.



(a) All clients in the peer-to-peer network are initially connected to each other.



(b) Example of an communication round. Client U receiving model parameters from a fraction of the neighbors after they have trained their model on local data. In this case model parameters are received from one neighbor V.

Figure 4.5: A general overview of the FedavgP2P experiments.

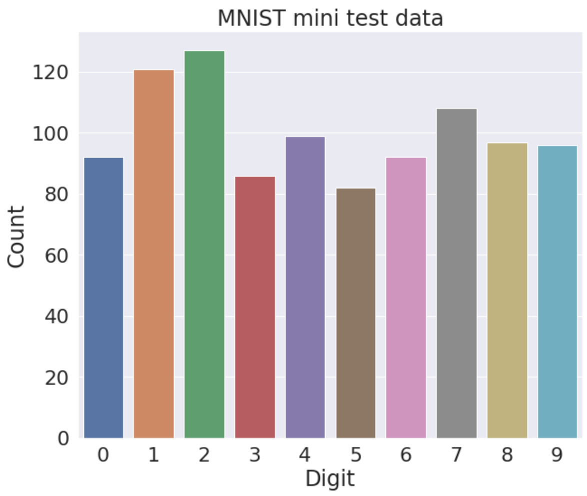


Figure 4.6: The mini MNIST test data set used for the heuristics that are based on per-class F1-scores.

4.5 Evaluation

4.5.1 Model evaluation

When evaluating ML models, it is common to divide data sets into a training set and test set. By evaluating the model on held-out samples, i.e. the test data, an unbiased estimate of a model's performance can be made [4]. In all experiments, we evaluated all the models on the test data based on the metric accuracy. Except for the experiments with the heuristics that are based on F1-scores, other metrics such as precision, recall, and F-score were not considered because they are more suitable to use when there are class imbalances. The MNIST data set is fairly balanced, therefore the accuracy metric was appropriate.

In the centralized experiments, we evaluated the aggregated averaged global model on the test set after every communication round. In the peer-to-peer FL experiments, we evaluated every client's model after every 10th communication round. This was done every 10th round due to being computationally expensive since we had to test 100 client models on the test data compared to one global model in the centralized FL experiments. Since there were 100 individual models in the peer-to-peer FL experiments, the variation of the models' accuracies was also studied. Thus, we examined the models' accuracies for all 100 clients every 10th communication round. In the peer-to-peer experiments, we also calculated an average of the model accuracy. The formula for calculating this follows:

$$Model\ average\ accuracy = \frac{\sum_{k=1}^{K} ModelTestAcc(w^k)}{K} \tag{4.1}$$

Where K represents the number of clients in the network, and the function *ModelTestAcc* calculates the model test accuracy with the given model weights w^k. In our conducted experiments $K = 100$.

4.5.2 Communication costs

Following McMahan et al. [16], we saved the communication round number where a target model accuracy of 97% had been reached in all experiments. The target model accuracy was used as a reference point that enabled us to compare communication costs in different experiments. To calculate the communication cost, we counted the number of models sent in the network for each experiment at the round where a model accuracy of 97% had been reached. In Fedavg, this included the number of models sent by each client and the central server, which was calculated as follows:

$$\text{Models sent in the network (Centralized FL)} = R * C * K * 2 + K \tag{4.2}$$

Where R refers to the round, C the fraction of clients the central server communicates with, and K the number of clients in the network.

In the FedavgP2P experiments, we counted the total of all models sent by each client when 97% average model accuracy had been reached, calculated as:

$$\text{Models sent in the network (P2P FL)} = R * C * A * K \tag{4.3}$$

Where R refers to the round and C the fraction of neighbors a client communicates with. The symbol A is the number of neighbors a client has, which is the same for all since it is a complete graph ($A = 99$ in all experiments). Finally, K is the number of clients in the network.

5 Experimental Results

In this chapter, we first compare how Fedavg and FedavgP2P affect the model convergence behavior in Section 5.1. In Section 5.2 we compare the communication costs of Fedavg and FedavgP2P. Further, in Section 5.3, we analyze the connection between local epochs, the fraction of neighbors a client communicates with, and how that affects model convergence and communication costs. Finally, in Section 5.4 we compare the different proposed heuristics for FedavgP2P.

5.1 Model convergence behavior

In this section, we cover the results of the models' convergence behaviors in the experiments with both IID and non-IID client data. In Figure 5.1 the results of Fedavg and FedavgP2P are presented for the experiments with local epochs (E) set to 5. For the other experiments with $E \in \{1, 10, 20\}$ see Appendix A. We generally see that the average model convergence behavior of FedavgP2P is comparable to Fedavg. Furthermore, the models' convergence behaviors are more stable with IID client data compared with non-IID data. Note that comparing Fedavg to FedavgP2P with the same C is not entirely fair due to the nature of the different algorithms. This is further discussed in Section 6.1.1.

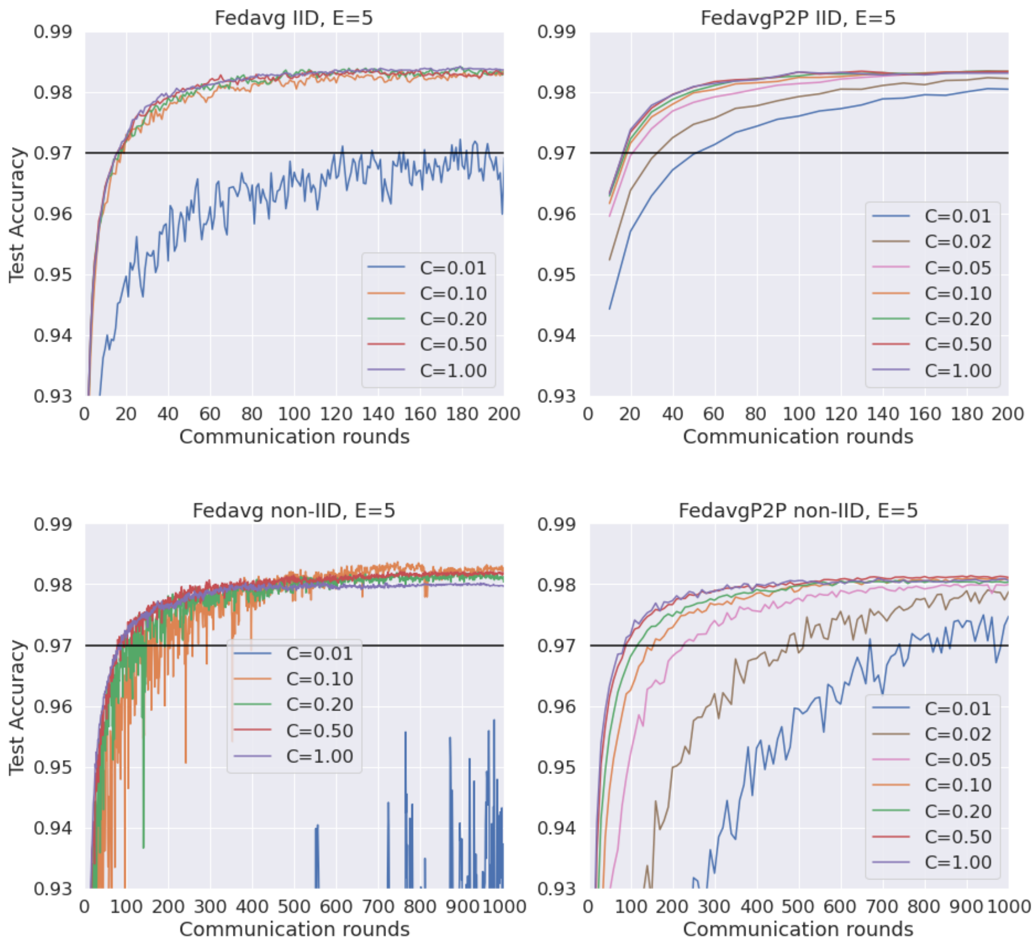


Figure 5.1: The results from the Fedavg and FedavgP2P experiments. Note that in the FedavgP2P figure, the average model accuracy is shown. *E* is the number of local epochs, which was set to 5. *C* is the fraction of clients the central server (or every client in the FedavgP2P experiments) had received updates from each round.

5.2 Communication costs

In Figures 5.2 and 5.3, and in Tables 5.1 and 5.2, we observe the number of models sent in the network at the round when the target model accuracy has been reached in the Fedavg and FedavgP2P experiments. In the Figures, we generally see that FedavgP2P requires a considerably more amount of models sent in the network to reach the target model accuracy compared to Fedavg. Considering the lowest values of models sent in the network (see Tables 5.1 and 5.2), we see that for the experiments with IID client data, Fedavg requires 336 models to be sent vs FedavgP2P's 5,000. This amounts to approximately 14.9x more communication with FedavgP2P. Furthermore, in the experiments with non-IID client data, the corresponding values are 67,000 vs. 2,160 which amounts to 31.0x more communication with FedavgP2P.

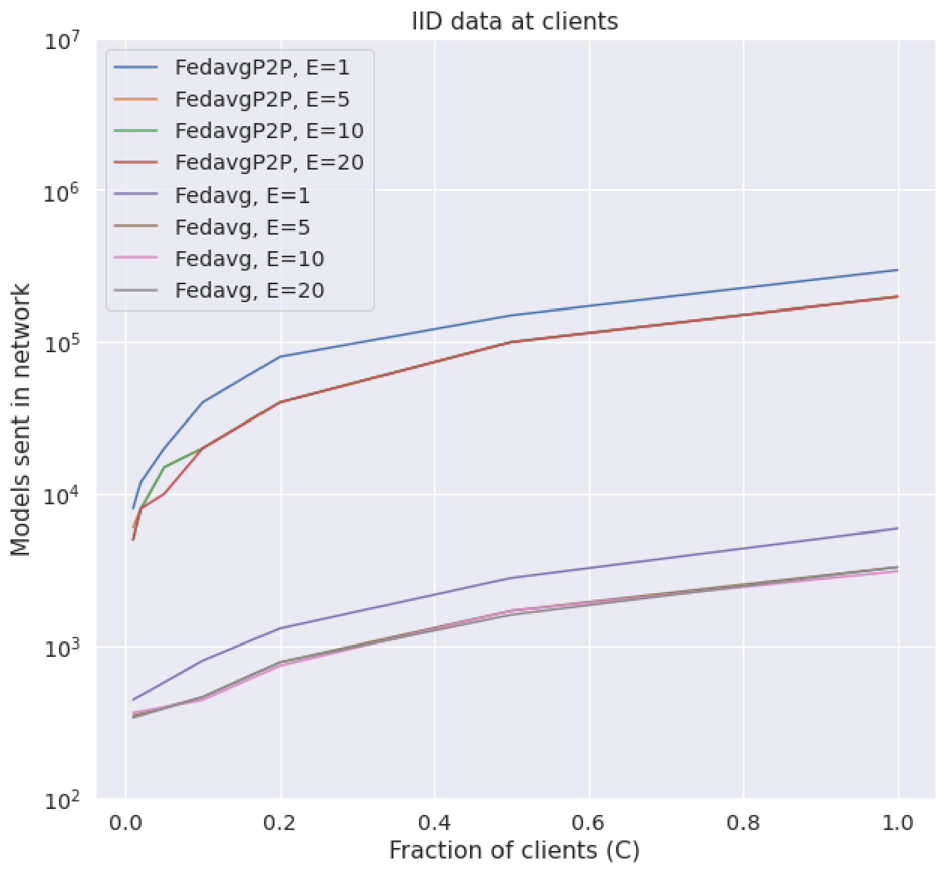


Figure 5.2: A comparison of Fedavg to FedavgP2P considering models sent in the network when 97% model accuracy had been reached. Note that in the FedavgP2P experiments, the values are given when 97% average model accuracy had been reached. IID client data. *E* is the number of local epochs and *C* is the fraction of clients the central server (or every client with FedavgP2P) had received updates from each round.

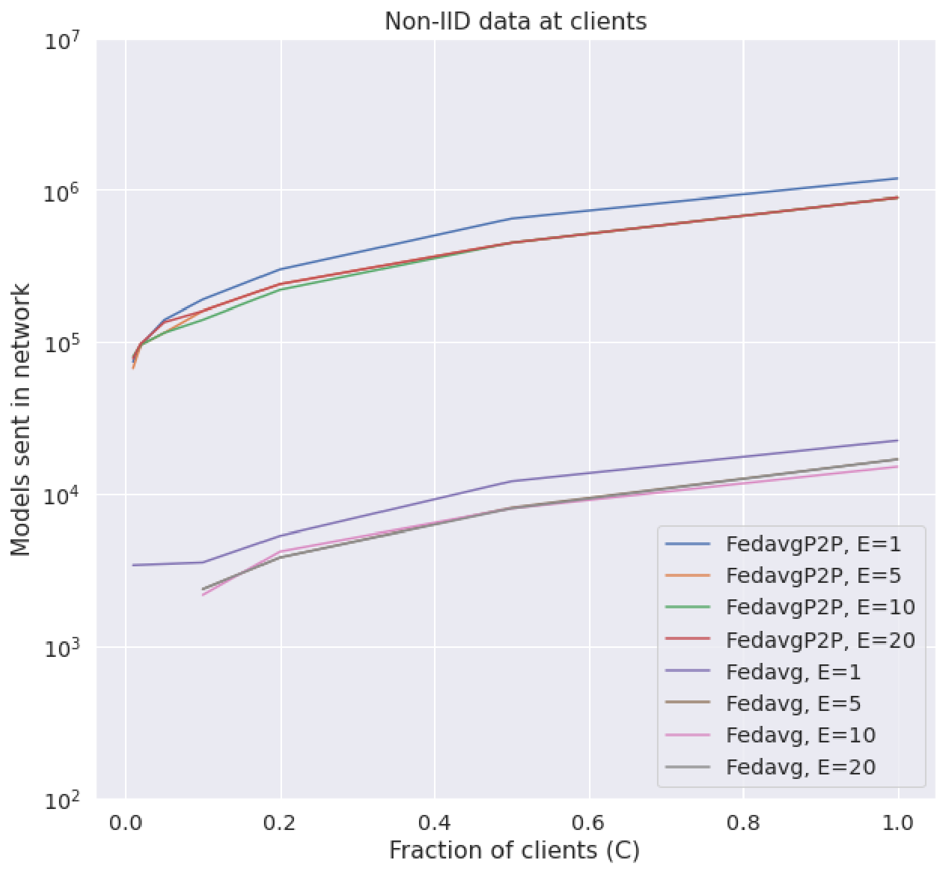


Figure 5.3: A comparison of Fedavg to FedavgP2P considering models sent in the network when 97% model accuracy had been reached. Note that in the FedavgP2P experiments, the values are given when 97% average model accuracy had been reached. Non-IID client data. *E* is the number of local epochs and *C* is the fraction of clients the central server (or every client with FedavgP2P) had received updates from each round.

Fedavg

E	C	Models sent in the network (IID)	Models sent in the network (non-IID)
1	0.01	442	3,398
1	0.10	800	3,540
1	0.20	1,300	5,300
1	0.50	2,800	12,100
1	1.00	5,900	22,500
5	0.01	346	(-)
5	0.10	460	2,360
5	0.20	780	3,820
5	0.50	1,700	8,100
5	1.00	3,300	16,900
10	0.01	362	(-)
10	0.10	440	2,160
10	0.20	740	4,180
10	0.50	1,700	8,000
10	1.00	3,100	15,100
20	0.01	336	(-)
20	0.10	460	2,360
20	0.20	780	3,820
20	0.50	1,600	8,000
20	1.00	3,300	16,900

Table 5.1: The communication costs of the Fedavg experiments when 97% model accuracy had been reached. E is the number of local epochs and C is the fraction of clients the central server had received updates from each round. Thus, in these experiments, the central server received updates from 1, 10, 20, 50, and 100 clients. Three of the experiments did not reach the target accuracy which is denoted by '(-)'.

FedavgP2P

E	C	Models sent in the network (IID)	Models sent in the network (non-IID)
1	0.01	8,000	74,000
1	0.02	12,000	96,000
1	0.05	20,000	140,000
1	0.10	40,000	190,000
1	0.20	80,000	300,000
1	0.50	150,000	650,000
1	1.00	297,000	1,188,000
5	0.01	6,000	67,000
5	0.02	8,000	96,000
5	0.05	15,000	115,000
5	0.10	20,000	160,000
5	0.20	40,000	240,000
5	0.50	100,000	450,000
5	1.00	198,000	891,000
10	0.01	5,000	80,000
10	0.02	8,000	96,000
10	0.05	15,000	115,000
10	0.10	20,000	140,000
10	0.20	40,000	220,000
10	0.50	100,000	450,000
10	1.00	198,000	891,000
20	0.01	5,000	78,000
20	0.02	8,000	98,000
20	0.05	10,000	135,000
20	0.10	20,000	160,000
20	0.20	40,000	240,000
20	0.50	100,000	450,000
20	1.00	198,000	891,000

Table 5.2: The communication costs of the FedavgP2P experiments when 97% average model accuracy had been reached. *E* is the number of local epochs and *C* is the fraction of neighbors each client had received updates from each round. Thus, in these experiments, for every client, updates were received from 1, 2, 5, 10, 20, 50, and 99 neighbors.

5.3 The effect of local epochs and fraction of neighbors

The effect of the number of local epochs (E) can be seen in Figure 5.4. In general, the results show that when increasing E from 1 to 5, fewer communication rounds (and thus lower communication cost) were needed to reach 97% model accuracy in almost all experiments. Increasing E further tended to give no substantial improvements, and in some cases, it instead resulted in more communication rounds than with lower E.

In Figure 5.5, the variation of the models' convergence behaviors can be observed for the FedavgP2P experiments with $E = 5$ and IID client data. The corresponding results with non-IID client data can be found in Figure 5.6. For the other experiments with $E \in \{1, 10, 20\}$ see Appendix A. As can be seen in Figure 5.5, as the fraction of neighbors a client communicates with (C) increases, the lower the variation of the models' convergence behaviors. Generally, we see that in the experiments with non-IID data, the variation is higher than with IID data.

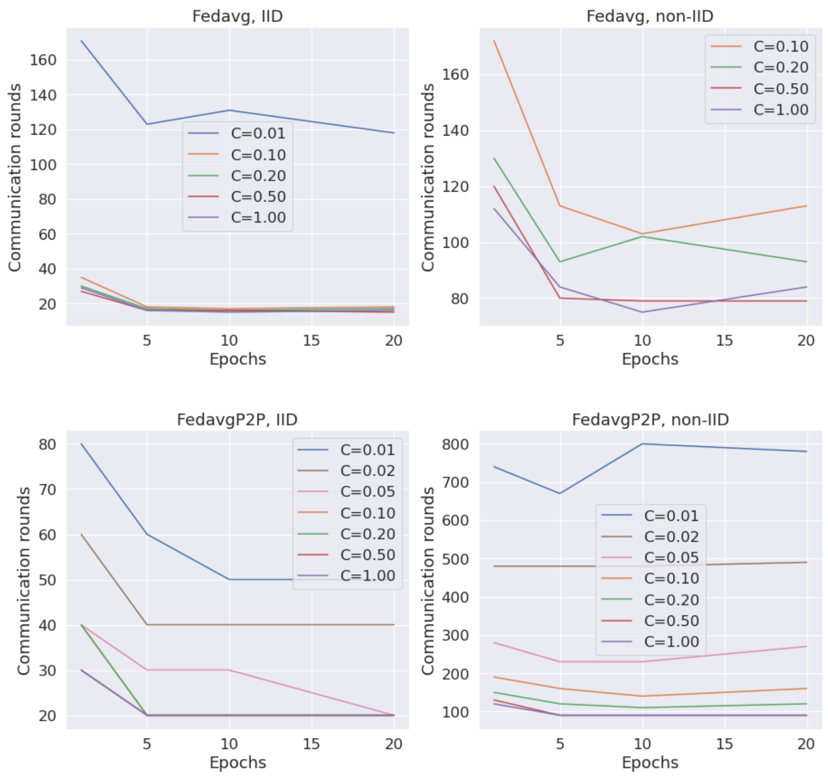


Figure 5.4: The impact of the number of local epochs. The communication round values are given when 97% model accuracy had been reached. Note that in the FedavgP2P experiments, the values are given when 97% average model accuracy had been reached. Results with both Fedavg and FedavgP2P with IID and non-IID client data are shown. C is the fraction of clients the central server (or each client in the FedavgP2P case) had received updates from each round.

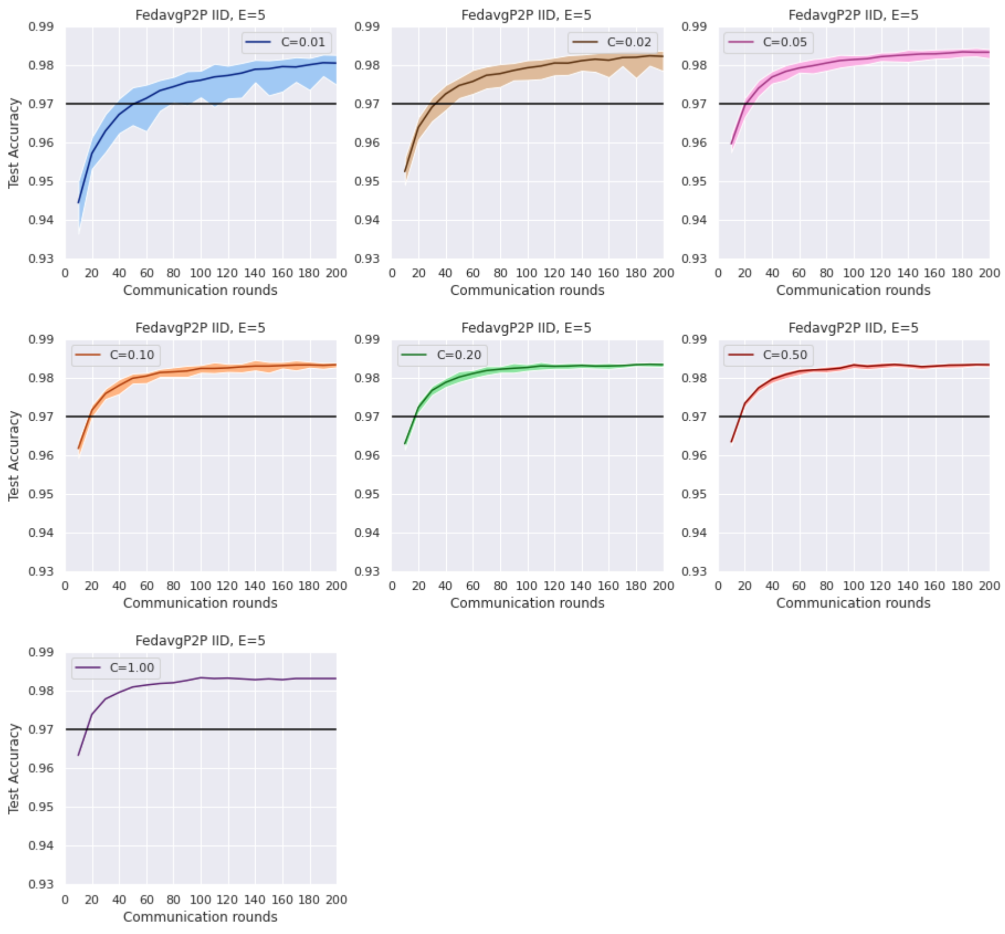


Figure 5.5: The variation of FedavgP2P models' accuracies during training with IID client data. The number of local epochs (E) was 5. The mean is represented by the darker lines. The outer edges of the pastel colors represent the values of the clients with the highest and lowest accuracy at that round. Thus, all 100 models' accuracies are within the colored areas.

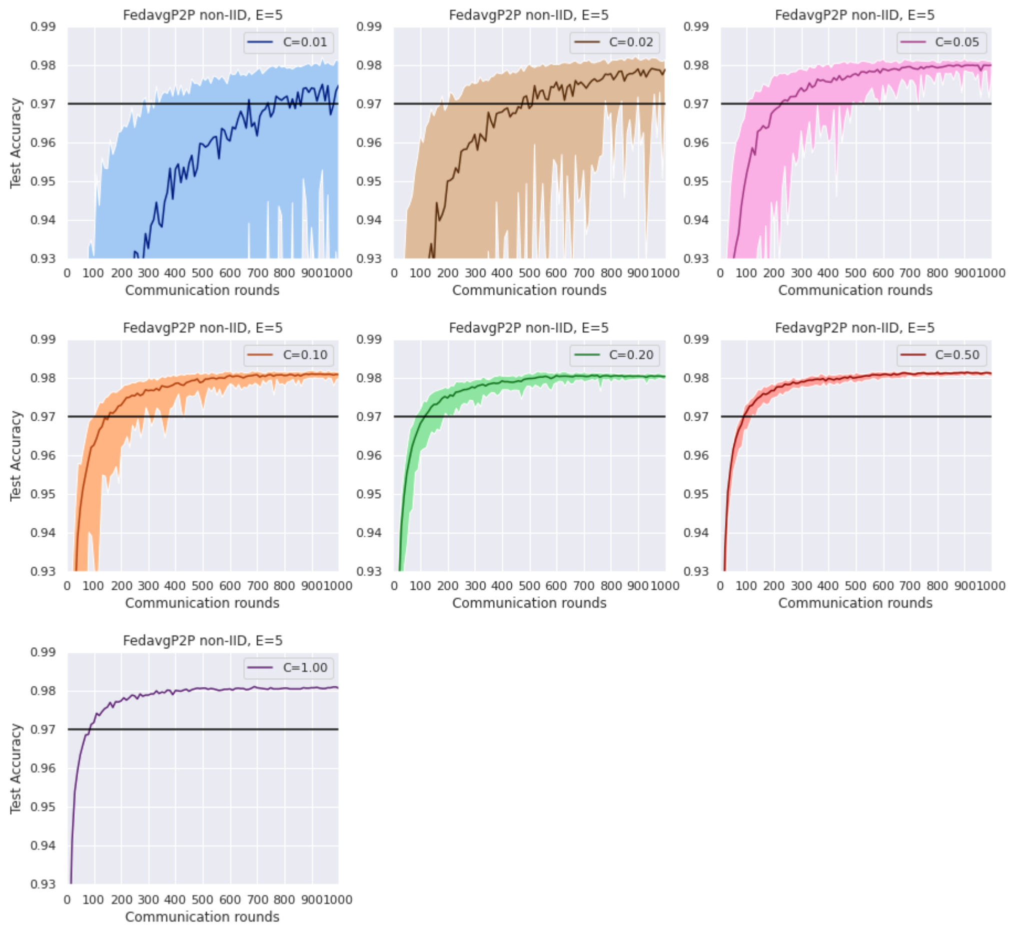


Figure 5.6: The variation of FedavgP2P models' accuracies during training with non-IID client data. The number of local epochs (E) was 5. The mean is represented by the darker lines. The outer edges of the pastel colors represent the values of the clients with the highest and lowest accuracy at that round. Thus, all 100 clients' model accuracies are within the colored areas

5.4 FedavgP2P with heuristics

In this section, we present the results of the FedavgP2P experiments with the three heuristics. In Figure 5.7 we can observe models' convergence behaviors. Surprisingly, in most cases, FedavgP2P with heuristics show more unstable convergence behavior compared to the original FedavgP2P. Figure 5.8 and 5.9 show the variation of the models' accuracies, there we see that the variation tends to be more dramatic with the heuristics. Lastly, in Figure 5.10 we can observe the number of models sent in the network when 97% average model accuracy has been reached in the various experiments. The heuristics *10 latest* and *F1-score arithmetic* showed that for some values on C slightly fewer models sent in the network to reach 97% average model accuracy were needed compared to FedavgP2P.

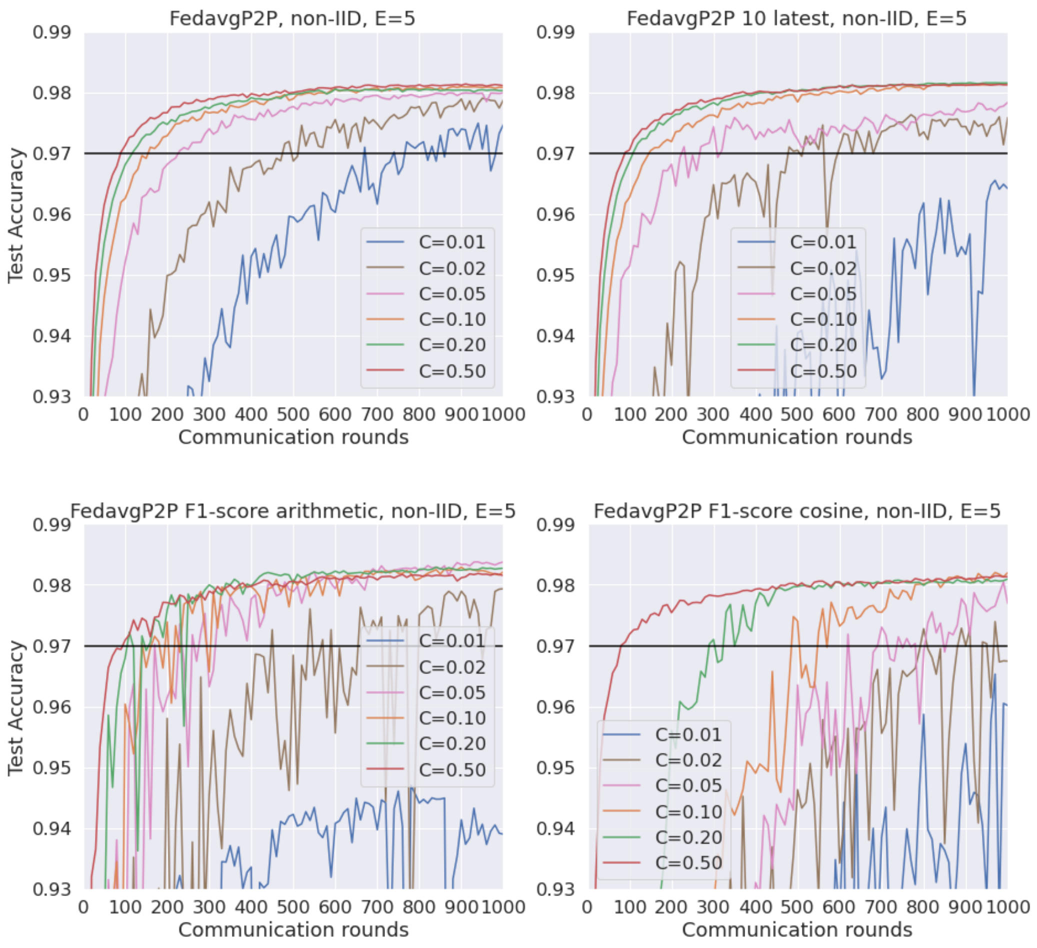


Figure 5.7: The results of FedavgP2P and FedavgP2P with heuristics. The model average accuracy is shown after every 10th communication round. Non-IID client data. E is the number of local epochs which was set to 5. C is the fraction of neighbors each client had received updates from each round. Thus, in these experiments, each client received updates from 1, 2, 5, 10, 20, 50 neighbors.

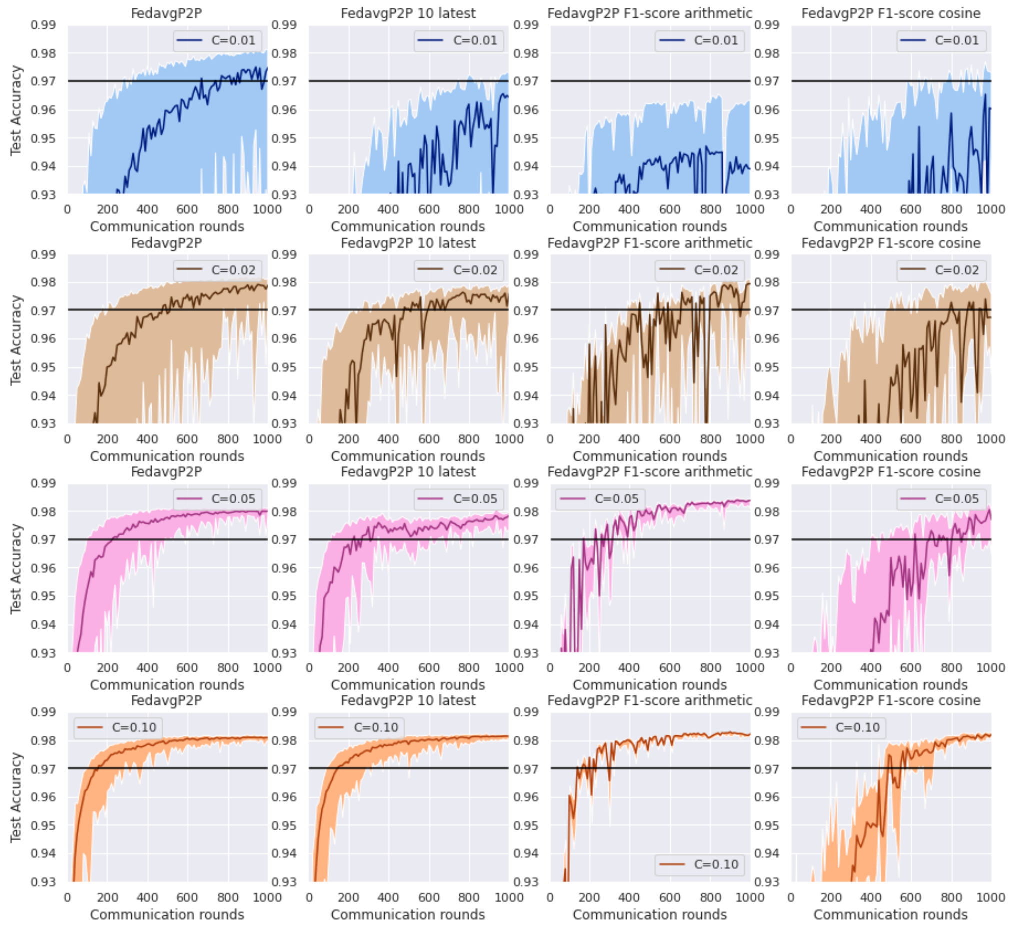


Figure 5.8: The variation of the models' accuracies with FedavgP2P and FedavgP2P with heuristics during training with non-IID client data. Local epochs (E) were set to 5. Fraction of neighbors (C) was set to 0.01, 0.02, 0.05 and 0.10. The mean is represented by the darker lines. The outer edges of the pastel colors represent the values of the clients with the highest and lowest accuracy at that round. Thus, all 100 models' accuracies are within the colored areas.

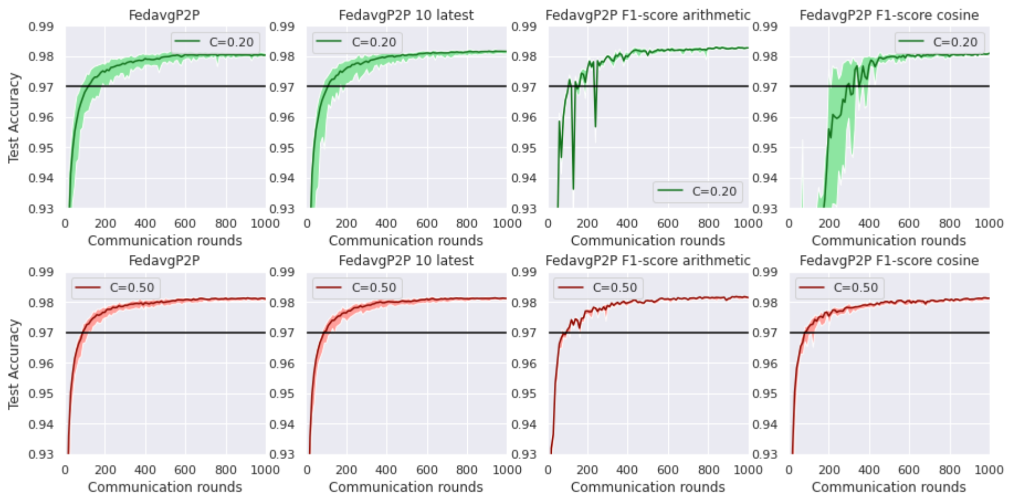


Figure 5.9: The variation of the models' accuracies with FedavgP2P and FedavgP2P with heuristics during training with non-IID client data. Local epochs (E) were set to 5. The fraction of neighbors (C) was set to 0.20 and 0.50. The mean is represented by the darker lines. The outer edges of the pastel colors represent the values of the clients with the highest and lowest accuracy at that round. Thus, all 100 models' accuracies are within the colored areas.

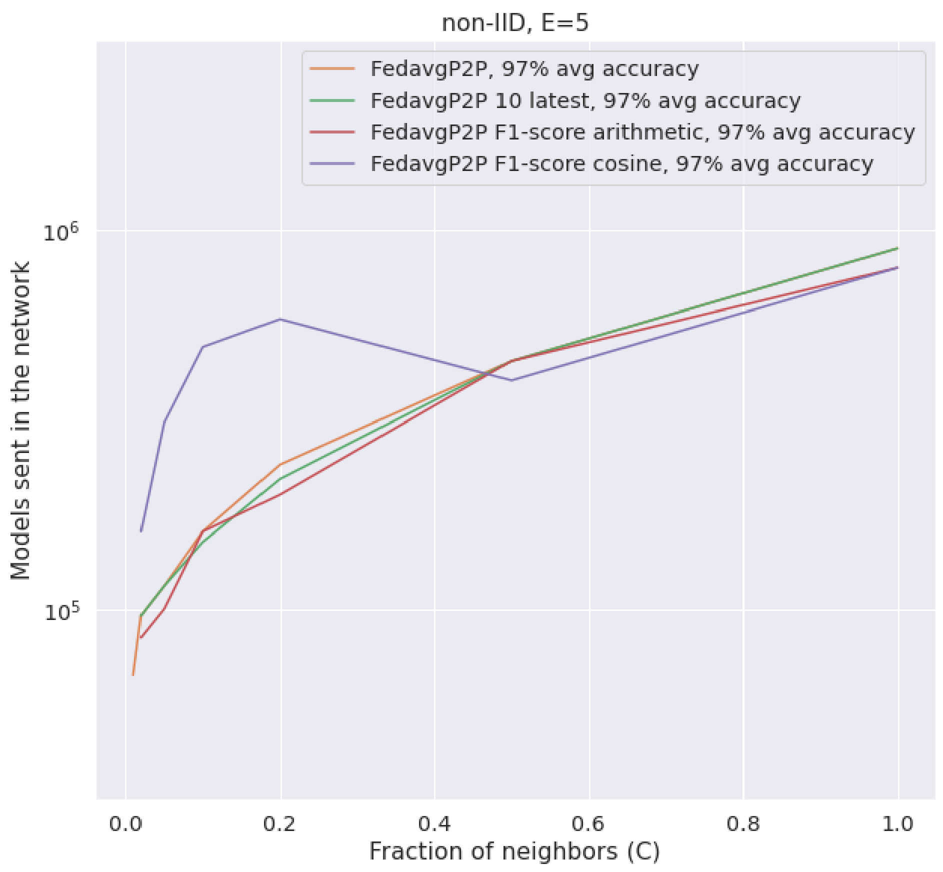


Figure 5.10: A comparison of the number of models sent in the network when 97% average model accuracy had been reached. Results from FedavgP2P and FedavgP2P with heuristics are shown. The number of local epochs E was 5 and C is the fraction of neighbors every client had received updates from each round.

6 Discussion

In this chapter, we discuss and analyze the results. Additionally, the method and the work in a wider context are discussed.

6.1 Results

6.1.1 Comparing FedavgP2P to Fedavg

Considering models' convergence behaviors, the results indicate that models trained with FedavgP2P can achieve comparable behaviors as Fedavg when training DNNs with both IID and non-IID client data. See Figure 5.1. However, FedavgP2P comes with the expense of a variation in models' accuracies and higher communication costs compared to Fedavg. This can be observed in Figures 5.2, 5.3, 5.5, and 5.6.

When observing the models' convergence behaviors for Fedavg and FedavgP2P in figures 5.1, we see that the general behaviors are quite similar for both methods. Most experiments end with around 98% models' accuracy. However, what cannot be seen in these Figures is the variation of the models' accuracies in FedavgP2P. This variation can be observed in Figures 5.5 and 5.6 when there are IID and non-IID client data. There, we see that as the fraction of neighbors (C) rises, the lower variation of the models' accuracies. With non-IID client data, when $C \leqslant 0.10$, the variation of models' accuracies is far greater and more unstable than in the experiments with $C > 0.10$. These results imply that the FedavgP2P average models' convergence behaviors are more comparable to Fedavg when C is of sufficient size.

Let us consider the experiments with the least amount of models sent in the network when 97% model accuracy had been reached. In both cases with IID and non-IID client data, FedavgP2P showed to require more overall network communication costs. However naturally with FedavgP2P, the communication cost burden is more evenly spread. But is the communication burden less per client in FedavgP2P than for the central server in Fedavg? By utilizing the equations 4.2 and 4.3, we can study this by comparing the number of sent models for the central server with Fedavg versus the sent models per client with FedavgP2P. With non-IID client data, the sought values are 1,180 for the central server versus 670 per client. In general, this means that with FedavgP2P compared to Fedavg, the overall network communication costs are higher, but the costs are more evenly spread among the participating clients rather than being heavily focused on a central server. Hence, if there is a communication constraint at the central server, e.g., insufficient bandwidth, FedavgP2P can be an appropriate approach. Further discussion of this follows in Subsection 6.1.4.

The fraction of neighbors (C) is 0.01 in the experiments with the lowest number of sent models needed to reach the target accuracy. An important aspect is thus the high variation of the models' accuracies in FedavgP2P when C is low. Comparing the average models' accuracies with FedavgP2P to Fedavg's model's accuracy when C is low is not a fair comparison due to high variation. Perhaps a more fair comparison between the two algorithms would be to compare the communication costs when all 100 models in the FedavgP2P experiments have reached 97% accuracy. If we instead compare in this manner, the FedavgP2P experiments with $C = 0.01$ would be invalid since all 100 models do not reach 97% in the given amount of rounds in the experiments. This means that the discrepancy between the communication costs of FedavgP2P and Fedavg is even larger with the new comparison method. Therefore, we can conclude that regardless of the comparison method, FedavgP2P needs considerably more communication costs in the network to reach comparable results with Fedavg.

6.1.2 The effect of local epochs and fraction of neighbors

When observing Figure 5.4, we see in almost all experiments that increasing the number of local epochs (E) from 1 to 5 reduces the total communication needed to reach 97% average model accuracy. Higher E further reduced the communication costs slightly in some experiments, and in others, it resulted in greater communication costs. For example, in the non-IID scenarios with $C = 0.01$, the number of communication costs needed to achieve 97% average accuracy decreased when the number of epochs changed from 1 to 5. However, communication costs increased when E was 10 and 20. A possible explanation for this is that at some point the models start overfitting their data if E is high enough, thus if the number of local epochs is above a certain level it could have a negative effect. This could explain why most benefits were gained when increasing E from 1 to 5. An alternative explanation could be that with high values of E, the trained models become too different considering the model parameters, hence an averaging of the models would not have a desirable effect due to them being too dissimilar.

The fraction of neighbors (C) used in each round affects the models' convergence behaviors as seen in Figure 5.5 and 5.6. With lower values of C, we observe less stable convergence behaviors which are demonstrated by the variety of the models' accuracies. However, we see that the behaviors are quite similar for $C = 0.2$, and $C = 0.5$. The number of models sent in these cases when 97% average model accuracy had been reached is 240,000 and 450,000. This implies that by tuning C, communication costs can be reduced while still maintaining relatively stable convergence behaviors in all models with FedavgP2P.

6.1.3 FedavgP2P with heuristics

If we study Figure 5.8, we observe that for lower values on the fraction of neighbors (C), the experiments with FedavgP2P heuristics show poorer convergence behaviors than with the original FedavgP2P. In general, considering the three FedavgP2P heuristics, the average model accuracy is more unstable and the variation is higher than the original FedavgP2P. However, as seen in Figure 5.9, for higher values of C, we observe comparable convergence behaviors for all algorithms. This could be explained by the high values of C since that leads to each client communicating with more neighbors every round. This indicates that when communicating with a large fraction of the clients in the network, the choice of neighbors each client communicates with does not matter.

Why did our heuristics not perform better than FedavgP2P? One reason could be that the heuristics cause the clients in the network to communicate with the same type of neighbors more frequently. This could in turn introduce several clusters in the network, where clients are more likely to communicate with neighbors in the same cluster. Furthermore, this could prolong the time for clients to receive model parameters from neighbors not belonging to the same cluster, which could result in inferior performance since it may reduce the diversity in the model parameters each client receives. As future work, it would be interesting to study if these clusters arise by analyzing the choice of neighbors for each client throughout the training process.

Surprisingly, when observing Figure 5.10, we see that FedavgP2P with heuristics in some experiments require less communication costs to reach 97% average model accuracy than original FedavgP2P. However, the improvements are not substantial, and there are no clear evidence that the improvements would indicate a reduction in communication costs compared to the original FedavgP2P in other experiments. Instead, the observed reduced communication costs could solely be explained by the random factor since the behavior was very unstable with the heuristics.

In conclusion, our results from the FedavgP2P experiments with three heuristics indicate no improvements in models' convergence behaviors or a reduction in communication costs.

6.1.4 When is FedavgP2P faster than Fedavg, considering training time?

Do our results indicate that there might be cases when FedavgP2P is faster than Fedavg, considering training time? The answer to this question depends on many aspects, such as communication constraints and clients' systems. For example, using Fedavg could be a faster approach if the central server has sufficient bandwidth. However, using FedavgP2P could also be faster if the central server does not have sufficient bandwidth.

To demonstrate cases where FedavgP2P can be faster than Fedavg, we utilize our experimental results in fictive scenarios where we assume that the central server bandwidth is the bottleneck with Fedavg. Similarly, the clients' bandwidths are the bottlenecks with FedavgP2P. Other costs such as computation at the clients are assumed to be negligible in these scenarios. We further assume that the central server can receive and send models at the same rate per time unit, thus we can calculate the time it takes to reach 97% model accuracy as a function of the bandwidth:

$$time_{centralized}(bandwidth_{server}) = \frac{sentModels_{centralized}}{bandwidth_{server}} \tag{6.1}$$

Where $bandwidth_{server}$ is the number of models the server can receive and send per time unit. $sentModels_{centralized}$ refers to the number of models sent in the network when 97% model accuracy has been reached.

In a similar way, we can calculate the time it takes to reach 97% average model accuracy as a function of the clients' bandwidths:

$$time_{p2p}(bandwidth_{client}) = \frac{sentModels_{p2p}}{K * bandwidth_{client}} \tag{6.2}$$

Here, we assume two characteristics 1) bandwidths are equal in all clients, and 2) the clients receive an equal amount of models every round concurrently. Furthermore, $bandwidth_{client}$ is the number of models a client can receive and send per time unit. $sentModels_{p2p}$ refers to the number of models sent in the network when 97% average model accuracy had been reached. K is the number of clients in the network, note that dividing by K follows from assumption 2.

FedavgP2P is thus faster than Fedavg if the following holds:

$$time_{p2p}(bandwidth_{client}) < time_{centralized}(bandwidth_{server}) \tag{6.3}$$

Using the Equations 6.1 and 6.2 in Statement 6.3 we get:

$$\frac{sentModels_{p2p}}{K * sentModels_{centralized}} < \frac{bandwidth_{client}}{bandwidth_{server}} \tag{6.4}$$

By utilizing Statement 6.4 and our experimental results, we can find all scenarios where FedavgP2P reach 97% average model accuracy faster than Fedavg to reach 97% model accuracy. For example, if we consider the results with non-IID client data, $C = 0.01$ and $E = 5$ for both FedavgP2P and Fedavg (see Table 5.1 and 5.2), FedavgP2P is faster than Fedavg when this holds:

$$\frac{160,000}{100 * 2,360} < \frac{bandwidth_{client}}{bandwidth_{server}} \approx 0.68 < \frac{bandwidth_{client}}{bandwidth_{server}} \tag{6.5}$$

Furthermore, we can also see that when it holds that $(0.68 < \frac{bandwidth_{client}}{bandwidth_{server}} < 1)$, it implies that FedavgP2P can be faster than Fedavg when the clients' bandwidths are slower than the central server bandwidth.

In these fictive scenarios, we assumed bandwidths as bottlenecks and thus could find when FedavgP2P is faster than Fedavg. In other scenarios, perhaps computation time at the clients are the bottleneck. Time can then be calculated as a function of client computation times. We can extend this even further and find equations and statements when time is dependent on both bandwidths and client computation times. In conclusion, FedavgP2P can be faster than Fedavg, which is partly shown in this section. However, when this happens depends on several aspects such as the communication constraints and clients' systems. Moreover, the experimental results from this thesis can further be taken into consideration for analyzing if FedavgP2P can be an appropriate choice in a certain domain.

6.2 Method

One limitation of the study is that the training of DNNs using the MNIST data set [12] could be considered not to be particularly challenging. Reasons for this are the simple NN architecture, the small grayscale images (28x28 pixels), and the few classes. Because of the simplicity of the problem, it is not clear that similar results attained in this thesis would also be achieved in more challenging scenarios. However, the results from this thesis give indications to how FedavgP2P could perform in more difficult problems. Moreover, by studying this problem, numerous experiments could be conducted due to them not being significantly computationally heavy. This enabled us to do a greater comparison and have time to study the heuristics.

The experiments in this thesis do not consider any discrepancy regarding client systems hardware and network connections. In essence, the experiments simulate a network with very ideal settings. Furthermore, all clients performed the same number of local epochs before sharing their local model parameters. This means that every client waits for all clients in the network to finish their local computation. In a real-world scenario, the waiting could potentially slow down the whole training process due to stragglers. This could be very impractical in some domains since it could prolong the training process considerably. In a federated scenario, there are likely to be varying kinds of client systems and network connections. To avoid long waiting for clients, it could be suitable for clients in the network to send models at the same time as others perform local computation. Another alternative is to allow for different amount of local computation before sharing model parameters similarly to the Fedprox algorithm [14] (see Section 2.4.2). Aspects considering systems and network heterogeneity could in future work be taken into account to understand FedavgP2P more thoroughly in more realistic scenarios.

Considering the algorithms, there is randomness involved. Thus there are concerns regarding the reliability of some of the experiments with non-IID client data. As previously discussed in the results, when the fraction of neighbors (C) is low, the variation of the models' convergence behaviors is high. Thus, the number of communication rounds needed to reach 97% average model accuracy could be different when running the experiments again. If more time and resources were available, a more preferable approach would be to run each experiment several times and average the results. Moreover, due to computational loads, models are only tested every 10th round in the FedavgP2P experiments. Because of the sparse testing and the variation of the models' convergence behaviors with low C, the results from those experiments work as rough general estimates.

The experiments in this thesis are highly replicable. The data set used is open for the public (see Section 4.2). Moreover, the software used is open source (see Section 4.1). Because of this, a practitioner can replicate the same experiments conducted in this thesis if satisfactory hardware is available.

6.2.1 Source criticism

The sources used in this thesis have been collected through the search engine *Google scholar*. Furthermore, the vast majority of the sources are peer-reviewed papers, which were chosen to ensure that the information used in this thesis are valid and reliable. Most of the work in this thesis builds upon the study from McMahan et al [16]. This includes the Fedavg algorithm, ways of evaluating the algorithms, and how to distribute the MNIST data set across clients. The work from McMahan et al is well-cited (1250+) and the authors are well established within the FL area, hence the source is reliable.

6.3 The work in a wider context

FL has the potential to enable wider adoption of ML in domains where data sharing is troublesome due to privacy and regulatory reasons. An example of a domain is healthcare, where FL is a promising approach that can utilize healthcare data to train ML models which assist, e.g., clinicians in their work [21]. Furthermore, efficiency can be improved and costs can be reduced with further adoption of ML in healthcare [26, 18]. The societal impact can be substantial, and FL can enable ML models to be trained with data that previously have been hard to utilize.

There are numerous ethical and societal challenges that are important to discuss with the wider adoption of FL. One challenge is: How do we ensure that ML models draw fair conclusions? Bias in data leads to ML models being unfair since the models just learn the patterns in the data [5]. If the raw training data is never shared, which can be the case when FL is used, it is challenging to find bias in the data since the full training data, in some cases, cannot be examined. Thus, there is a risk that ML models trained with FL can draw biased conclusions that can lead to detrimental consequences. It is therefore of the utmost importance to ensure that ML models work as intended considering fairness and reliability before we use them in critical parts of society.

7 Conclusion

This chapter summarises the thesis. In Section 7.1, we cover how the aim was achieved and present answers to the research questions. Further, in Section 7.2, we present an overall thesis conclusion. Finally, proposals of future work are presented in Section 7.3.

7.1 Aim and research questions

The overarching aim of this thesis was to study the viability of peer-to-peer FL. To do this we utilized the presented algorithm FedavgP2P. Comparisons of FedavgP2P were made to the centralized counterpart Fedavg. A total of 114 experiments were conducted, each experiment consisted of 100 simulated clients in a network. The goal of the clients in the network was to collaboratively train DNNs for the task of digit recognition using the MNIST data set [12]. Furthermore, the experiments were analyzed to answer the research questions. The answers to the questions follow in the subsections below.

7.1.1 How does FedavgP2P compare to Fedavg concerning models' convergence behaviors with IID and non-IID client data?

FedavgP2P can achieve comparable convergence behavior to Fedavg. However, this comes with a variation in the models' convergence behaviors. When there are IID client data, the variation is lower than with non-IID data. Furthermore, the variation is strongly affected by the number of neighbors each client communicates with each round. When the number of neighbors rises, the variation decreases and the models' convergence behaviors become more stable and more comparable to Fedavg.

7.1.2 How does FedavgP2P compare to Fedavg concerning communication costs with IID and non-IID client data?

Comparing the results that had the least amount of communication costs with IID client data, FedavgP2P required 14.9x more communication than Fedavg to reach a target model accuracy. Considering scenarios with non-IID client data, FedavgP2P required 31.0x more communication. However, the FedavgP2P experiments with the lowest communication costs also came with a great variation in models' convergence behaviors.

With FedavgP2P, naturally, the communication costs are more evenly distributed and not heavily focused on a central server as with Fedavg. This makes the proposed method scale better than Fedavg in cases where the number of clients is increased. Furthermore, we found that even though the overall communication costs are higher with FedavgP2P, faster training times could be achieved if there are communication constraints at a central server with Fedavg (e.g., insufficient bandwidth).

7.1.3 Considering FedavgP2P: how does the number of local epochs, and the number of neighbors each client communicates with every round affect the models' convergence behaviors and communication costs?

Increasing the number of local epochs up to a certain level reduced communication costs to reach a target average model accuracy notably. Increasing the number of local epochs further tended to give no substantial improvements, and in some cases, it instead resulted in higher communication costs. One plausible explanation could be that the models start to overfit the local data when there is too much local computation every round. The number of neighbors each client communicates with every round affects the stability and variation of the models' convergence behaviors. Communicating with more neighbors led to lower variation but increased the communication costs in the network needed to reach a target model average accuracy.

7.1.4 How can FedavgP2P be enhanced such that models' convergence behaviors improve and communication costs decrease?

We attempted to improve FedavgP2P with three different heuristics that utilize client identities and per-class F1-scores. Clients chose neighbors to communicate with every round based on the client identities and F1-scores as opposed to purely random in FedavgP2P. Unfortunately, our experiments did not show any evidence that the heuristics performed better than the original FedavgP2P. Possibly, the heuristics cause a reduction of the diversity in model parameters each client receives. However, further analysis is required to understand why inferior performance was observed.

7.2 Thesis conclusion

The overall findings presented in this thesis indicate that peer-to-peer FL could be a viable approach to train DNNs collaboratively across multiple clients without them sharing raw training data. First and foremost, the results show that models trained with FedavgP2P are comparable to models trained with its centralized counterpart Fedavg. As evident in the results, FedavgP2P comes with larger overall network costs compared to Fedavg since more data has to be transmitted to achieve comparable models' convergence behaviors. However, by utilizing a peer-to-peer topology, several benefits such as no single point of failure and no central-server dependency are achieved. This makes peer-to-peer FL a strong choice if these characteristics are required. Further, since there is no central-server dependency, we found that faster training times of the models can be attained using FedavgP2P compared to Fedavg if the central server has communication constraints such as insufficient bandwidth. In conclusion, our results of peer-to-peer FL, through FedavgP2P, show that it is a promising approach to train DNNs across multiple clients.

7.3 Future work

As future work, it would be interesting to see how FedavgP2P performs in more challenging problems and data sets. For example, in a network with more than 100 clients. One suitable data set could thus be FEMNIST [2], which was created to mimic a federated scenario with approximately 3500 users with non-IID data. Similarly, the proposed heuristics could be studied in conjunction with FEMNIST to find if they could improve FedavgP2P in a more challenging problem. Furthermore, a natural extension of this thesis would be to study systems heterogeneity among clients. For instance, this could be simulated by computing different amounts of work for every client. Future work could also be to further analyze the experimental results and find more scenarios where FedavgP2P could be an appropriate choice. For example, this could be to find when FedavgP2P is faster than Fedavg when there are weak client systems. Finally, further privacy measures could be considered in future work with FedavgP2P to minimize the risk of raw data being exposed more. This could be achieved by utilizing methods such as differential privacy [17].

Bibliography

[1] Christopher Briggs, Zhong Fan, and Peter Andras. "Federated learning with hierarchical clustering of local updates to improve training on non-IID data". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–9.

[2] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečn, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. "Leaf: A benchmark for federated settings". In: *arXiv preprint arXiv:1812.01097* (2018).

[3] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. "The secret sharer: Evaluating and testing unintended memorization in neural networks". In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 267–284.

[4] Rich Caruana and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 161–168.

[5] Alexandra Chouldechova and Aaron Roth. "A snapshot of the frontiers of fairness in machine learning". In: *Communications of the ACM* 63.5 (2020), pp. 82–89.

[6] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. "Dermatologist-level classification of skin cancer with deep neural networks". In: *nature* 542.7639 (2017), pp. 115–118.

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[8] Lie He, An Bian, and Martin Jaggi. "COLA: Decentralized Linear Learning". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018, pp. 4536–4546. URL: https://proceedings.neurips.cc/paper/2018/file/05a70454516ecd9194c293b0e415777f-Paper.pdf.

[9] István Hegedűs, Gábor Danner, and Márk Jelasity. "Decentralized learning works: An empirical comparison of gossip learning and federated learning". In: *Journal of Parallel and Distributed Computing* 148 (2021), pp. 109–124.

[10] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. "Advances and open problems in federated learning". In: *arXiv preprint arXiv:1912.04977* (2019).

[11] Jakub Konečn, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. "Federated learning: Strategies for improving communication efficiency". In: *arXiv preprint arXiv:1610.05492* (2016).

[12] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[13] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated learning: Challenges, methods, and future directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60.

[14] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated Optimization in Heterogeneous Networks". In: *Proceedings of Machine Learning and Systems*. Ed. by I. Dhillon, D. Papailiopoulos, and V. Sze. Vol. 2. 2020, pp. 429–450. URL: https://proceedings.mlsys.org/paper/2020/file/38af86134b65d0f10fe33d30dd76442e-Paper.pdf.

[15] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. "Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017, pp. 5330–5340. URL: https://proceedings.neurips.cc/paper/2017/file/f75526659f31040afeb61cb7133e4e6d-Paper.pdf.

[16] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 1273–1282.

[17] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. "Learning Differentially Private Recurrent Language Models". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=BJ0hF1Z0b.

[18] Bertalan Meskó, Gergely Hetényi, and Zsuzsanna Győrffy. "Will artificial intelligence solve the human resource crisis in healthcare?" In: *BMC health services research* 18.1 (2018), p. 545.

[19] Róbert Ormándi, István Hegedűs, and Márk Jelasity. "Gossip learning with linear models on fully distributed data". In: *Concurrency and Computation: Practice and Experience* 25.4 (2013), pp. 556–571.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[21] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. "The future of digital health with federated learning". In: *NPJ digital medicine* 3.1 (2020), pp. 1–7.

[22] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information processing & management* 45.4 (2009), pp. 427–437.

[23] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. "On unbiased sampling for unstructured peer-to-peer networks". In: *IEEE/ACM Transactions on Networking* 17.2 (2008), pp. 377–390.

[24] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. "Revisiting unreasonable effectiveness of data in deep learning era". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852.

[25] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. "Federated machine learning: Concept and applications". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19.

[26] Kun-Hsing Yu, Andrew L Beam, and Isaac S Kohane. "Artificial intelligence in healthcare". In: *Nature biomedical engineering* 2.10 (2018), pp. 719–731.

A Full Experimental Results

This appendix presents the experimental results more thoroughly. Section A.1 presents the model convergence behavior with Fedavg and FedavgP2P. Section A.2 covers the model variation in all FedavgP2P experiments.

A.1 Model convergence behavior

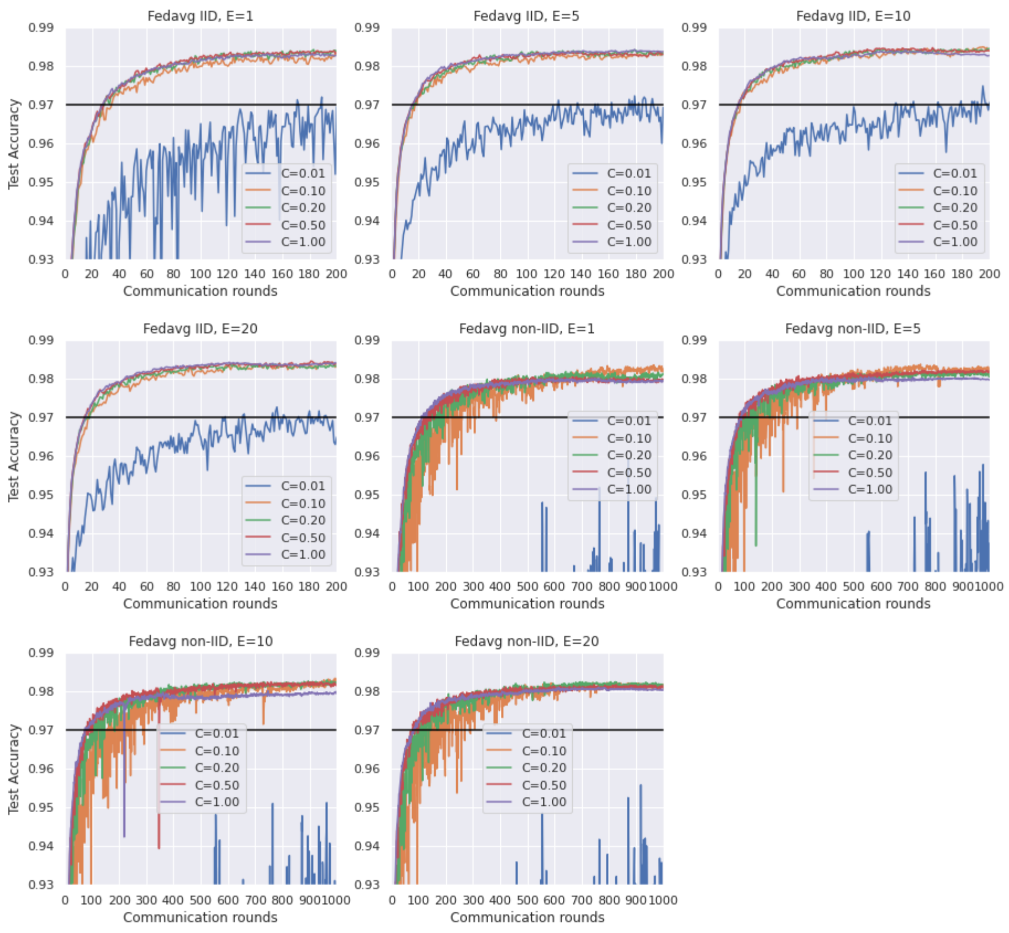


Figure A.1: The results of Fedavg experiments. Model accuracy is shown after every communication round. Results from IID and non-IID client data are presented. E is the number of local epochs on each client and C is the fraction of clients the central server had received updates from each round. Thus, in these experiments, the central server received updates from 1, 10, 20, 50, and 100 clients.

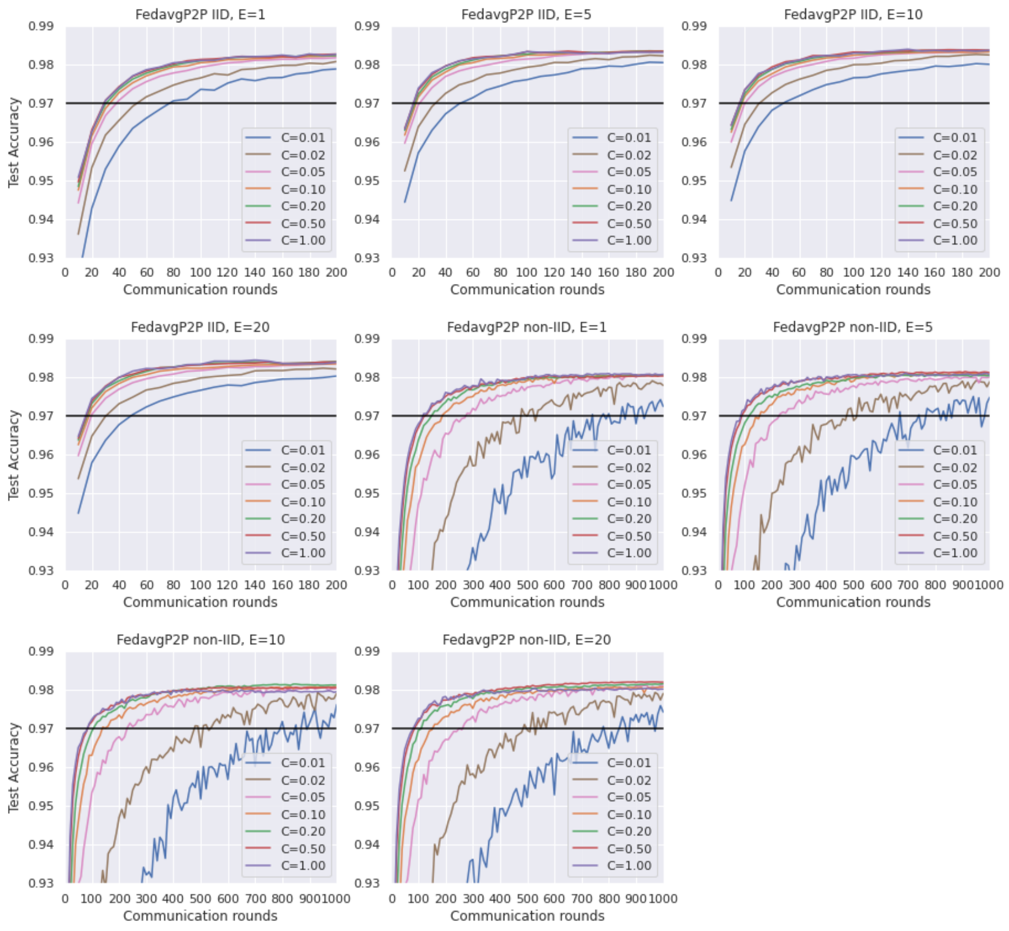


Figure A.2: The results of FedavgP2P experiments. The models average accuracy is shown after every 10th communication round. Results from IID and non-IID client data are presented. *E* is the number of local epochs on each client and *C* is the fraction of neighbors each client had received updates from each round. Thus, in these experiments, each client received updates from 1, 2, 5, 10, 20, 50 and 99 clients.

A.2 Model variation

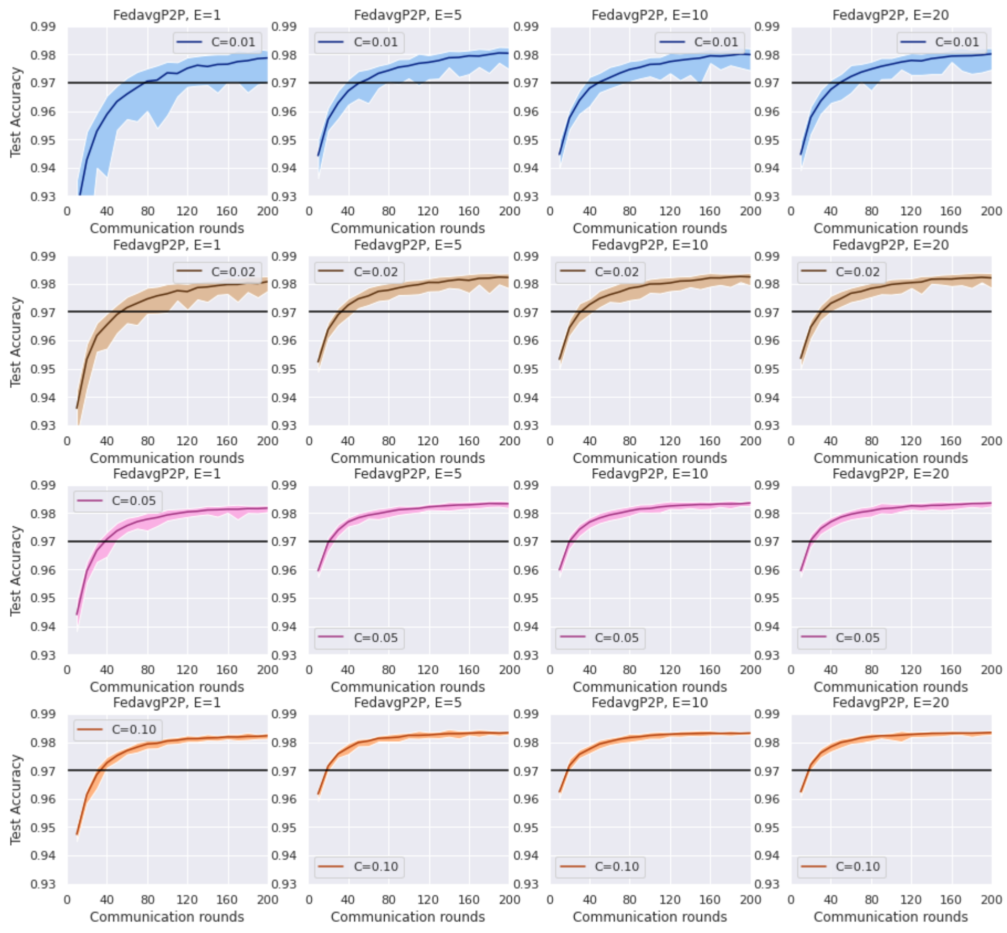


Figure A.3: The variation of FedavgP2P models' accuracies during training with IID client data. *E* is the number of local epochs. The mean is represented by the darker lines. The outer edges of the pastel colors represent the values of the clients with the highest and lowest accuracy that round. Thus, all 100 models' accuracies are within the colored areas.

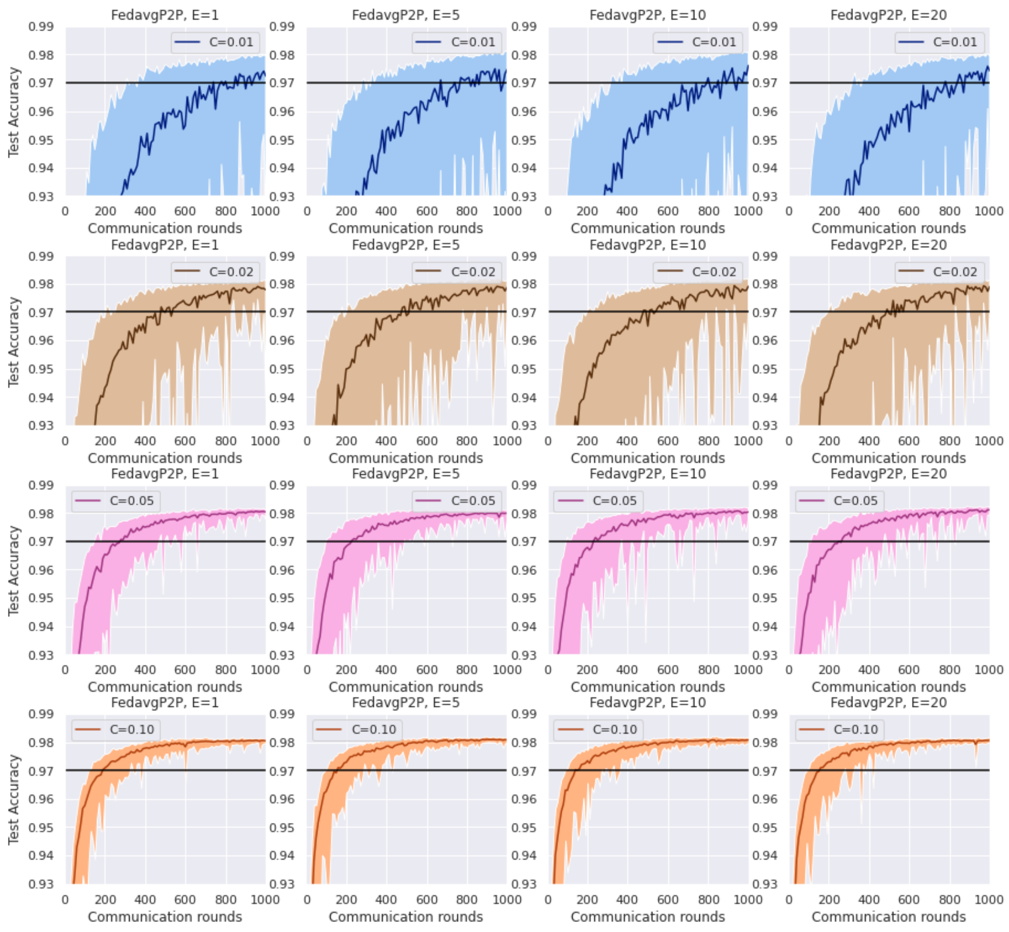


Figure A.4: The variation of FedavgP2P models' accuracies during training with non-IID client data. *E* is the number of local epochs. The mean is represented by the darker lines. The outer edges of the pastel colors represent the values of the clients with the highest and lowest accuracy that round. Thus, all 100 clients' model accuracies are within the colored areas