Article

Federated Learning: Centralized and P2P for a Siamese Deep Learning Model for Diabetes Foot Ulcer Classification

Mohammud Shaad Ally Toofanee, Mohamed Hamroun, Sabeena Dowlut, Karim Tamine, Vincent Petit, Anh Kiet Duong and Damien Sauveron

Abstract: It is a known fact that AI models need massive amounts of data for training. In the medical field, the data are not necessarily available at a single site but are distributed over several sites. In the field of medical data sharing, particularly among healthcare institutions, the need to maintain the confidentiality of sensitive information often restricts the comprehensive utilization of real-world data in machine learning. To address this challenge, our study experiments with an innovative approach using federated learning to enable collaborative model training without compromising data confidentiality and privacy. We present an adaptation of the federated averaging algorithm, a predominant centralized learning algorithm, to a peer-to-peer federated learning environment. This adaptation led to the development of two extended algorithms: Federated Averaging Peer-to-Peer and Federated Stochastic Gradient Descent Peer-to-Peer. These algorithms were applied to train deep neural network models for the detection and monitoring of diabetic foot ulcers, a critical health condition among diabetic patients. This study compares the performance of Federated Averaging Peer-to-Peer and Federated Stochastic Gradient Descent Peer-to-Peer with their centralized counterparts in terms of model convergence and communication costs. Additionally, we explore enhancements to these algorithms using targeted heuristics based on client identities and f1-scores for each class. The results indicate that models utilizing peer-to-peer federated averaging achieve a level of convergence that is comparable to that of models trained via conventional centralized federated learning approaches. This represents a notable progression in the field of ensuring the confidentiality and privacy of medical data for training machine learning models.

Keywords: centralized and peer-to-peer architecture; confidentiality of data; diabetes foot ulcer and deep learning; federated learning; Siamese network

1. Introduction

Diabetes represents an important health problem in the world, and the World Health Organization (WHO) encourages researchers to work along with this objective [1]. Approximately 537 million individuals are projected to rise to 783 million by 2045, according to research by the International Diabetes Federation (IDF) [2]. Diabetic foot ulcers (DFUs) are a prevalent complication among individuals with diabetes mellitus. Preventing and effectively treating diabetic foot ulcers is a challenging task due to their high recurrence rate. Recent advancements in the field of machine learning (ML) have led to highly effective innovations across various domains. For instance, in the specialized field of dermatology, a machine learning model has been employed to diagnose skin cancer and has achieved comparable results to
dermatologists [3,4]. This is true when dealing with digital medical images as well as textual health data, with the possibility of generating reports that extract quantitative, objective, structured, and personalized information from stroke MRIs with performance comparable to that of an expert evaluator.

Furthermore, many recent ML applications rely heavily on deep learning [5], which necessitates sufficiently large and diverse datasets to ensure reliability [6]. However, the collection of such datasets can be challenging. In many domains, data are owned by numerous clients and stored in various locations. Due to privacy and regulatory concerns, data sharing among clients is not possible. The issues associated with data sharing make it difficult to generate robust ML models. Consequently, the existing collected data are not fully leveraged by ML. This unfortunately negatively impacts the development of high-performing ML models. Robust ML models have the potential to enhance efficiency and reduce costs in numerous fields, and one such field that is of concern to us in this study is healthcare [7,8].

Applying deep learning model to DFU classification also requires massive, reliable data, which are unfortunately not readily available. They could be available if we can ensure that several healthcare centers participate to help build a robust model trained on massive data. While hospitals may be convinced by the potential of deep learning, but the main concern is of the confidentiality and privacy of the data. Federated learning is a method that allows clients to collaboratively train ML models without sharing raw training data [9]. Normally, ML models are trained in a centralized location, where the model owner can freely observe all the training data. However, in federated learning, model training is decentralized. The predominant FL strategy involves using a central orchestration server that distributes a global model to participating clients. These clients then train the models using their local data. The updated parameters of the local model are then sent to the central server, where the global model is updated by aggregating and combining the parameters from the clients’ models. In the industry, some large technology companies have adopted FL in production, and many startups intend to use FL to address regulatory and privacy concerns. However, FL poses several challenges, such as communication efficiency, system heterogeneity, non-identically distributed client data (non-IID), and privacy protection [10]. For example, non-IID client data, such as an imbalanced distribution of labels, can significantly impede the learning process [11].

With centralized FL, clients must trust and rely on a central server. This approach carries the risk of disrupting the training process in the event of server failure. Additionally, in FL scenarios, where the number of participating clients is potentially high, the central server must handle a large number of communications, which can be a limiting factor [11]. To address certain issues associated with centralized FL, peer-to-peer (P2P) FL could be a viable alternative, as it allows for bypassing dependence on the central server. To achieve this, we extend an important centralized FL algorithm called FedAvg [9] to operate in a P2P environment. This extension draws inspiration from other works that explore decentralized model training [12–14].

The motivation for this research is to evaluate the feasibility of implementing a deep learning-based classification of DFU using a peer-to-peer federated learning approach and to compare its efficacy with a centralized federated learning framework. Furthermore, while there has been significant research focused on the classification of DFU and the application of federated learning to publicly available datasets by developers, to the best of our knowledge, this study represents the first attempt to apply a federated learning approach specifically to DFU classification using a dataset obtained from the DFUC2021 [15] challenge organized by the Medical Image Computing and Computer-Assisted Intervention (MICCAI) society [16].

The novelty of this work lies in the exploration and adaptation of the FedAvg algorithm [9] for a peer-to-peer (P2P) setting, which we term FedAVGP2P, and the introduction of its counterpart, FedSGDP2P. Furthermore, this study is among the first to investigate the practicality of applying federated learning algorithms within a P2P architecture to the meticulously labeled DFU dataset. We explore its application for DFU in a novel architecture DFU-SIAM [17], which has yielded very promising results for DFU classification compared to other research in this field. We study two algorithms, which are FedAVGP2P and FedSGDP2P.
A comparative evaluation is conducted with an accuracy threshold of 90%. We focus on empirical performance metrics, such as model convergence and communication overhead in both IID and non-IID data distributions across local clients. Moreover, this research is pioneering in proposing and integrating novel heuristics to enhance the performance and efficiency of FedAVGP2P and FedSGDP2P, paving the way for experimenting with other heuristics for future federated P2P learning frameworks.

The subsequent sections of this paper are organized as follows: Section 3 provides a comprehensive analysis of relevant previous works in the field. In Section 4, we present a detailed description of our proposed solution. This is followed by the presentation of our experimental results in Section 5. Finally, we conclude the paper with a summary of our findings and suggestions for future research, which are presented in Section 6.

2. Background and Preliminaries

In this section, we explain the Siamese neural network and two distinct federated learning architectures: centralized federated learning and peer-to-peer federated learning architectures.

2.1. Siamese Neural Network (SNN)

A Siamese neural network is a category of neural network architectures that contain two or more identical subnetworks. By identical, we mean that they have the same structure, the same parameters, and the same weights. It was introduced by Bromley et al. [18] for signature verification written on a tablet. During training, the two sub-networks extract features from two signatures, while the joining neuron measures the distance between the two feature vectors. Verification consists of comparing an extracted feature vector with a stored feature vector of the person signing.

These subnetworks, which make up the Siamese neural networks, are constructed as feedforward perceptrons, and utilize error backpropagation during the training process. They work in parallel, comparing their output using the cosine distance as illustrated in Figure 1.

![Figure 1](image_url)  
**Figure 1.** Representation of the structure of the Siamese neural network model. The data are processed from left to right. The value of the cosine distance is a measure of the similarity between the input pair of data instances as the final output [19].
Deep neural networks (DNNs) are recognized for their reliance on extensive datasets for effective training. For instance, if a model is trained on 10 classes and an extra class is introduced later, the entire model necessitates retraining. In contrast, Siamese neural networks are distinguished for their one-shot learning capability. This signifies that the incorporation of a new class does not mandate a complete retraining of the model. One-shot learning teaches the model to make its own assumptions about their similarities based on the minimal number of visuals. There can be only one image or a very limited number of them, in which case it is often called few-shot learning for each class.

As an example, consider differentiating between dogs and cats. A traditional ML model would necessitate a large dataset of thousands of training examples [20], encompassing various angles, lighting conditions, and backgrounds. In contrast, one-shot learning defies the need for an extensive array of examples in each category. It harnesses its acquired knowledge from prior tasks of the same type, drawing connections among similar objects and effectively categorizing unfamiliar objects into their respective classes.

During the training of the SNN, we need to ensure two inputs:
1. The feature vectors of similar and dissimilar pairs should be descriptive, informative, and distinct enough from each other so that segregation can be learned effectively.
2. The feature vectors of similar image pairs should be similar enough, and those for dissimilar pairs should be dissimilar enough so that the model can quickly learn semantic similarity.

To ensure the model can learn similarity and dissimilarity, it uses a loss function called the contrastive loss function. The contrastive loss function is a distance-based loss function that updates weights such that two similar feature vectors have a minimal Euclidean distance. In comparison, the distance is maximized between two different vectors. The constrastive loss function is given in Equation (1) below:

\[
(1 - y) \frac{1}{2} (D_w)^2 + y \frac{1}{2} (\max(0, m - D_w))^2
\]

In Equation (1), \(y\) represents whether or not the vectors are dissimilar, and \(D_w\) is the Euclidean distance between the vectors. When the vectors are dissimilar (\(y = 1\)), the loss function minimizes the second term, for which \(D_w\) must be maximized (encourage more distance between dissimilar vectors). We want these vectors to have a distance of more than at least \(m\) (which is a Margin), and we avoid computation if the vectors are already \(m\) units apart by defaulting to zero.

2.2. Centralized Federated Learning Architecture

In centralized federated learning, there exists a centralized server that coordinates the whole training process.

The central server is responsible for the following task:
1. Determines a global model to be trained.
2. Selects participants (i.e., local nodes) for each training round.
3. Aggregates local training results sent by the participants.
4. Disseminates the updated model to the participants.
5. Terminates the training when the global model satisfies some requirements (e.g., accurate threshold is reached).

Figure 2 shows the mechanics of the centralized architecture. From the network perspective, we can immediately deduce that this architecture generates high communication costs between servers and clients and is also a vulnerable point of failure for the overall learning process.
2.3. Federated Learning: Peer-to-Peer Architecture

The architecture of federated learning based on peer-to-peer interaction operates without the need for a central server to coordinate the learning and parameter sharing process. Participants engage in direct communication without relying on an intermediary. This results in an equitable standing for each participant within the architecture, enabling any participant to initiate a model exchange request with others [22]. Due to the absence of a central server, participants must establish a prior consensus regarding the sequence in which models are to be transmitted and received.

Figure 2 illustrates the centralized FL architecture inspired by [21]. Step1: Participant selection and global model dissemination. Step2: Local computation. Step3: Local models aggregation. Step4: Global model update.

Figure 3 illustrates the P2P FL architecture inspired by [23]. It shows clients directly communicate with one another instead of any central authority. A group of clients with a common goal collaborate to improve their models by sharing information from peer to peer.

Figure 3. The P2P FL architecture inspired by [23].
When assessing vulnerabilities, the P2P FL architecture proves superior due to its avoidance of a central server, mitigating the risks associated with a single point of failure. Nonetheless, the efficiency of this approach can be influenced by the manner in which clients are interconnected [24], potentially impacting communication costs. Hence, achieving an equilibrium between performance and communication expenses becomes imperative within the P2P FL framework.

2.4. Federated Learning Algorithm

In federated learning, an aggregation algorithm refers to a technique implemented for consolidating the outcomes of training numerous intelligent models on the clients’ devices, utilizing their respective local datasets. This algorithm plays a crucial role in combining the results derived from the local client training processes and subsequently updating the global model [25]. Two such algorithms are:

1. Federated Stochastic Gradient Descent (FedSGD) averages the locally computed gradient at every step of the learning phase.
2. Federated averaging (FedAVG) averages local model updates when all the clients have completed training their models.

Before moving forward, we shall introduce some terms:

- **Round**: A round in federated learning is an iteration of the federated learning process. In each round, a subset of clients is selected to participate in the training process.
- **Clients**: k randomly selects a subset of K clients to participate in the current epoch.
- **Non-IID dataset**: This stands for a non-independent and identically distributed dataset. For an image classification problem, this means we may have some classes which exist for some clients but do not exist for another client. Non-IID poses a challenge to deep learning models, as it can lead to biased or unreliable models, resulting in low accuracy and incorrect results.
- **IID Dataset**: This stands for independent and identically distributed dataset. For an image classification problem, it means that each image has a similar probability distribution as the others, and all are mutually independent.

2.4.1. Federated Stochastic Gradient Descent (FedSGD)

FedSGD is an optimization algorithm used in federated learning (FL) to train machine learning models on decentralized data. It is a variation of the traditional Stochastic Gradient Descent (SGD) algorithm, adapted to the federated setting. FedSGD is a distributed version of SGD and uses the computation power of several compute nodes instead of one [26]. In FedSGD [27], the central model is distributed to the clients, and each client computes the gradients using local data. These gradients are then passed to the central server, which aggregates the gradients in proportion to the number of samples present on each client to calculate the gradient descent step.

The key difference between FedSGD, described in Algorithm 1, and traditional SGD lies in the aggregation step. In SGD, the local updates from all devices are typically averaged to update the global model. Moreover, a fraction of devices is randomly selected to participate in each round of model updates. This selective participation helps reduce the communication overhead and computational burden.

Since there is a need to send parameters to the main server after, each gradient calculation has a bandwidth cost; this may be a problem if the clients have limited connectivity access. This issue is tackled by federated averaging (FedAVG).
Algorithm 1 Federated Stochastic Gradient Descent (FedSGD) algorithm

1: **Input:**
2: Global model parameters: \( \theta_0 \)
3: Number of federated rounds: \( T \)
4: Learning rate for clients: \( \eta \)
5: **Initialization:**
6: Initialize global model parameters: \( \theta_0 \)
7: for \( t = 1 \) to \( T \) do
8: Select a subset of client devices: \( C_t \)
9: for each client \( i \in C_t \) in parallel do
10: Receive the current global model parameters: \( \theta_{t-1} \)
11: Sample a mini-batch of local data: \( B_i \)
12: Compute the local gradient: \( g_i \leftarrow \nabla f_i (\theta_{t-1}; B_i) \)
13: Update the client’s local model: \( \theta_{t_i} \leftarrow \theta_{t-1} - \eta \cdot g_i \)
14: end for
15: Aggregate local models to update the global model:
16: \( \theta_t \leftarrow \sum_{i \in C_t} \frac{|B_i|}{|B|} \cdot \theta_{t_i} \)
17: end for
18: **Output:** Final global model: \( \theta_T \)

2.4.2. Federated Averaging (FedAVG)

Federated averaging (FedAVG) is a communication-efficient algorithm for distributed training with an enormous number of clients [28]. It ensures data privacy and security and maintains data locality by enabling model training without sharing the raw data. It uses one aggregation by the server in each communication round, which significantly reducing the communication cost between the server and clients. Instead of sharing the gradients with the central server, weights tuned to the local model are shared. Finally, the server aggregates the clients’ weights (model parameters). The fundamental idea is that clients run multiple updates of model parameters before passing the updated weights to the central server [26]. Algorithm 2 describes the logic of FedAVG.

Algorithm 2 Federated averaging. The \( K \) clients are indexed by \( k \); \( B \) is the local minibatch size, \( E \) is the number of local epochs, and \( \eta \) is the learning rate [9]

1: Server executes:
2: Initialize \( w_0 \)
3: for each round \( t = 1, 2, \ldots \) do
4: \( m \leftarrow \max (C, K, 1) \)
5: \( S_t \leftarrow \) (random set of \( m \) clients)
6: for each client \( k \in S_t \) in parallel do
7: \( w_{t+1}^k \leftarrow \) (ClientUpdate) \( (k, w_t) \)
8: end for
9: \( w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{|B_k|}{|B|} \cdot w_{t+1}^k \)
10: end for
11: function ClientUpdate\((k, w)\) \( \triangleright \) Run on client \( k \)
12: \( \beta \leftarrow \) (split \( P_k \) into batches of size \( B \))
13: for each local epoch \( i = 1 \) to \( E \) do
14: for each batch \( b \in \beta \) do
15: \( w \leftarrow w - \eta \nabla l(w; b) \)
16: end for
17: end for
18: return \( w \) to server

2.4.3. Federated Averaging: Peer-to-Peer (FedAVG2P)

FedAVG2P is an extension or variation of the Federated Stochastic Gradient Descent (FedSGD) algorithm in federated learning. In the standard FedSGD algorithm, a central server
coordinates the federated learning process, where clients compute gradients on their local data and send them to the server for aggregation and model updates. In the FedSGDP2P variant, the communication process occurs directly between participating clients in a peer-to-peer manner, eliminating the need for a central server. Clients collaborate with each other to exchange gradient information and update their models collectively. This approach has the potential to enhance privacy, reduce communication overhead, and improve the scalability of federated learning. However, it may introduce challenges related to synchronization, security, and the management of peer-to-peer networks.

2.4.4. Federated SGD: Peer-to-Peer (FedSGDP2P)

FedAvgP2P is an extension or variation of the FedAVG algorithm in federated learning. In FedAvg, a central server coordinates the model aggregation process, where local models from participating clients are averaged to update a global model.

In the FedAvgP2P variant, the aggregation process involves peer-to-peer communication among participating clients, bypassing the need for a central server. Clients directly communicate with each other to exchange their local model updates and collectively compute the global model through decentralized means.

3. Related Work

This section investigates primarily the application of federated learning for the confidentiality of data.

In their latest article, Moshawrab et al. [25] review the use of federated learning and its application in the prediction of disease. They discuss the use of FL for diagnosing FL in the diagnosis of cardiovascular disease, diabetes, and cancer. Quite naturally, with the use of medical data, they stress the need for privacy and confidentiality when dealing with sensible data. They identify other areas, aside from healthcare, where the implementation of FL makes sense, including smart retail, transportation, natural language processing, and finance.

When dealing with FL, there is a need to strike a balance between performance and communication cost. Asad et al. [29] consequently evaluated the cost of communication efficiency in FL algorithms. They relied on latency and bandwidth as limitations and proposed the use of the Averaging Algorithm (FedAVG), Sparse Ternary Compression (STC), Communication-Mitigating Federated Learning (CMFL), and Federated Maximum and Mean Discrepancy (FedMMD) to evaluate communication efficiency. All the algorithms were evaluated on the CIFAR and MNIST datasets using a model that is convolutional neural network (CNN)-based. The data were divided in two ways to cater to the independent and identically distributed (IID) scenario and the non-IID scenario. The following parameters were used in the evaluation: client = 100, number of classes = 10, batch size = 20, and participation = 10%. Unfortunately, in this research, none of the algorithms were able to prove the best solution. However, the authors use this work to identify gaps and provide avenues for future research.

He et al. [12] introduced COLA, a decentralized training algorithm designed to optimize communication efficiency, scalability, and elasticity while also accounting for unreliable and heterogeneous devices to accommodate data changes, while Lin et al. [30] explored approaches for enhancing mini-batch stochastic gradient (SGD) algorithms and presented a novel post-local SGD method that achieves remarkable performance gains compared to training with large batches. These improvements were observed across well-known benchmark datasets, all while ensuring efficiency and scalability. Roy et al. [31] introduced a fully decentralized architecture called P2P FL (peer-to-peer federated learning) to overcome the limitations of classical federated learning. The conventional federated learning approach involves a centralized controller that collects and consolidates training data from all nodes, maintaining a global model on a cloud-based infrastructure across the network. However, the P2P FL architecture deploys nodes throughout the network, allowing them to interact exclusively with their immediate neighbors, thus eliminating the necessity of a centralized controller. This development in P2P federated learning enables nodes to engage with their next-hop neighbors in just two steps.
While federated learning (FL) presents a paradigm shift towards preserving data confidentiality, it is not without its challenges and limitations. One significant concern is the delicate balance between performance and communication efficiency. Asad et al. [29] pointed out that despite employing various algorithms aimed at enhancing communication efficiency, none proved to be the ultimate solution. This suggests that there is a substantial trade-off between algorithmic performance and communication overhead. Furthermore, the reliance on datasets like CIFAR and MNIST, which are relatively simplistic, may not adequately represent the complexity of real-world data, especially in non-IID scenarios, where data distribution is imbalanced across nodes. Moreover, the literature reflects a gap in addressing the complexities of managing sensitive medical data, where the stakes for privacy and accuracy are notably high. Collectively, these weaknesses support the need for ongoing research to refine FL algorithms, enhance their robustness, and ensure they are applicable to the dynamic and complex nature of real-world problems.

4. Proposed Methods

In FedAVG, a centralized server is mandatory for taking care of all transactions. By referring to previous research on decentralized training algorithms [12–14], we enhance FedAVG to operate within a peer-to-peer framework, thereby eliminating the necessity for a central server. We further extend our study by applying another variation of federated learning, which is FedSGD [32].

The extended algorithms are referred to as FedAVGP2P and FedSGDP2P. Each client has their own model and communicates directly with other clients. Before training, all client models are initialized with the same weights. Each client performs training on the model using its local data. Then, each client aggregates and averages updates from a set of random neighbors or selected users using a heuristic. This process is repeated for a finite number of rounds, allowing each client to have a fully trained global model without relying on a central server. A similar distributed computation is performed by the FedSGDP2P algorithm: during each round, clients calculate the gradient derived from the loss function on their local data. These gradients are then sent to other selected clients (either randomly or based on heuristics) to aggregate them and update the parameters of their models. Similar to FedAVG, FedAVGP2P and FedSGDP2P have four hyperparameters: the fraction of neighbors from which each client receives updates, the size of the local minibatch, the number of times each client trains on the shortest time period, the number of times each client trains on the local dataset in each round (epochs), and the learning rate.

4.1. Heuristic 0: Random

This approach is performed in a naive manner, where we simply perform random sampling. In other words, each client randomly sends its weight/vector gradient to a subset of other clients.

1. FedAVG (Algorithm 3)

**Algorithm 3** FedAVG Heuristic 0: random. $c$ is the fraction of clients that perform a computation on each round

1: for round ← 1, 2, 3, ... do
2: for client ← 1, 2, 3, ... do
3: \[ w_{client} = \text{fit}(w_{client}, \text{data}_{client}, \text{epochs} = $local$\_epoch) \]
4: end for
5: for client ← 1, 2, 3, ... in parallel do
6: \[ w_{client} = \text{Mean(GetRandomNeighbors(c).weight)} \]
7: end for
8: end for
2. FedSGD (Algorithm 4)

**Algorithm 4** FedSGD Heuristic 0: random. \( c \) is the fraction of clients that perform a computation on each round

```plaintext
1: for round ← 1, 2, 3, . . . do
2:   for local_epoch ← 1, 2, 3, . . . do
3:     for step ← 1, 2, 3, . . . do
4:       for client ← 1, 2, 3, . . . in parallel do
5:         grad\_client = Gradient(w\_client, data\_client)
6:         grad = getRandomNeighborsGrad(c)
7:         w\_client += Mean(grad)
8:       end for
9:     end for
10: end for
```

In the original FedAVGP2P algorithm, the selection of neighbors for communication is performed randomly. In order to enhance the performance of FedAVGP2P, we propose three distinct heuristics for choosing the neighbors to communicate with.

4.2. Heuristic 1: n Lastest

Each client in the network maintains its own identity and keeps track of the identities of the \( n \) most recent clients it has interacted with. At the end of each communication round, this information regarding the \( n \) most recent clients is disseminated throughout the network. Subsequently, each client selects its communication partners based on the level of dissimilarity in their previous interactions. Specifically, clients prioritize communication with those who have had the least amount of overlap in past interactions.

1. FedAVG (Algorithm 5)

**Algorithm 5** FedAVG Heuristic 1: n lastest. \( c \) is the fraction of clients that perform a computation on each round

```plaintext
1: for round ← 1, 2, 3, . . . do
2:   for client ← 1, 2, 3, . . . do
3:     w\_client = fit(w\_client, data\_client, epochs = $local\_epoch)
4:   end for
5: for client ← 1, 2, 3, . . . in parallel do
6:   neighbors = GetRandomNeighbors(c, without = client.last)
7:   w\_client ← Mean(neighbors.weight)
8:   client.last = (client.last + neighbors)[-n:]
9: end for
10: end for
```
2. FedSGD (Algorithm 6)

Algorithm 6 FedSGD Heuristic 1: \( n \) lastest. \( c \) is the fraction of clients that perform a computation on each round

\begin{algorithm}
\begin{algorithmic}[1]
\State for round \( \leftarrow 1,2,3,\ldots \) do
\State \hspace{1em} for local\_epoch \( \leftarrow 1,2,3,\ldots \) do
\State \hspace{2em} for step \( \leftarrow 1,2,3,\ldots \) do
\State \hspace{3em} for client \( \leftarrow 1,2,3,\ldots \) in parallel do
\State \hspace{4em} grad\_client = Gradient(w\_client, data\_client)
\State \hspace{4em} neighbors = GetRandomNeighbors(c, without = client.last)
\State \hspace{4em} w\_client\_+ = Mean(neighbors.grad)
\State \hspace{4em} client.last = (client.last + neighbors)[-n:]
\State \hspace{3em} end for
\State \hspace{2em} end for
\State \hspace{1em} end for
\State end for
\end{algorithmic}
\end{algorithm}

4.3. Heuristic 2: F1 Score

The second and third heuristics utilize the models’ performances to promote communication between clients with better-performing or dissimilar models. After each round, clients calculate their models’ per-class f1-scores on a test set and share them with the network. Clients then select neighbors to communicate with based on the dissimilarity or similarity scores computed using these f1-scores.

\begin{equation}
\text{neighbor dissimilarity score} = ndc = \sum_{\text{class } i} \left| F_{i}^{k} - F_{i}^{c} \right|
\end{equation}

1. FedAVG (Algorithm 7)

Algorithm 7 FedAVG Heuristic 2: F1 score

\begin{algorithm}
\begin{algorithmic}[1]
\State for round \( \leftarrow 1,2,3,\ldots \) do
\State \hspace{1em} for client \( \leftarrow 1,2,3,\ldots \) do
\State \hspace{2em} w\_client = fit(w\_client, data\_client, epochs = $local\_epoch$)
\State \hspace{2em} end for
\State \hspace{1em} end for
\State end for
\end{algorithmic}
\end{algorithm}

2. FedSGD (Algorithm 8)

Algorithm 8 FedSGD Heuristic 2: F1 score. \( c \) is the fraction of clients that perform a computation on each round

\begin{algorithm}
\begin{algorithmic}[1]
\State for round \( \leftarrow 1,2,3,\ldots \) do
\State \hspace{1em} for local\_epoch \( \leftarrow 1,2,3,\ldots \) do
\State \hspace{2em} for step \( \leftarrow 1,2,3,\ldots \) do
\State \hspace{3em} for client \( \leftarrow 1,2,3,\ldots \) in parallel do
\State \hspace{4em} grad\_client = Gradient(w\_client, data\_client)
\State \hspace{4em} neighbors = GetRandomNeighbors(c, without = client.last, metric = ndc)
\State \hspace{4em} w\_client\_+ = Mean(neighbors.grad)
\State \hspace{3em} end for
\State \hspace{2em} end for
\State \hspace{1em} end for
\State end for
\end{algorithmic}
\end{algorithm}
4.4. Heuristic 3: Score Cosine

For heuristic 3 clients select neighbors to communicate with based on the dissimilarity or similarity scores obtained using cosine score. Dissimilarity is calculated using Equation (3).

\[
\text{neighbor dissimilarity score} = \text{ndc} = \cos(F_k, F_c) = \frac{F_k \cdot F_c}{\|F_k\| \cdot \|F_c\|}
\] (3)

1. FedAVG (Algorithm 9)

Algorithm 9 FedAVG heuristic3: Score cosine. \( c \) is the fraction of clients that perform a computation on each round

1: \( \text{for} \ \text{round} \leftarrow 1, 2, 3, \ldots \ \text{do} \)
2: \qquad \text{for} \ \text{client} \leftarrow 1, 2, 3, \ldots \ \text{do} \)
3: \qquad \ \ \ \ \text{\( w_{\text{client}} = \text{fit}(w_{\text{client}}, data_{\text{client}}, \text{epochs} = \$local\_epoch) \)}
4: \qquad \text{end for}
5: \text{for} \ \text{client} \leftarrow 1, 2, 3, \ldots \ \text{in parallel do}
6: \qquad \text{neighbors} = \text{GetNeighbors}(c, \text{without} = \text{client.last}, \text{metric} = \text{ndc})
7: \qquad \text{\( w_{\text{client}} \leftarrow \text{Mean}(\text{neighbors.weight}) \)}
8: \text{end for}
9: \text{end for}

2. FedSGD (Algorithm 10)

Algorithm 10 FedSGD heuristic3: Score cosine. \( c \) is the fraction of clients that perform a computation on each round

1: \( \text{for} \ \text{round} \leftarrow 1, 2, 3, \ldots \ \text{do} \)
2: \qquad \text{for} \ \text{local\_epoch} \leftarrow 1, 2, 3, \ldots \ \text{do} \)
3: \qquad \quad \text{for} \ \text{step} \leftarrow 1, 2, 3, \ldots \ \text{do} \)
4: \qquad \qquad \text{for} \ \text{client} \leftarrow 1, 2, 3, \ldots \ \text{in parallel do} \)
5: \qquad \qquad \quad \text{\( \text{grad}_{\text{client}} = \text{Gradient}(w_{\text{client}}, data_{\text{client}}) \)}
6: \qquad \qquad \quad \text{neighbors} = \text{GetNeighbors}(c, \text{without} = \text{client.last}, \text{metric} = \text{ndc})
7: \qquad \qquad \quad \text{\( w_{\text{client}}+ = \text{Mean}(\text{neighbors}.\text{grad}) \)}
8: \qquad \qquad \text{end for}
9: \qquad \text{end for}
10: \text{end for}
11: \text{end for}

5. Experiments and Results

5.1. Experimental Setup

The experimental setup was conducted on a Windows 10 Pro operating system, running on a powerful hardware configuration comprising 64 GB of RAM and an Intel(R) Xeon(R) W-2155 CPU operating at 3.30 GHz. The system was further enhanced with an NVIDIA GeForce RTX 3060 GPU, boasting 12 GB of dedicated memory. To facilitate the experiments, the system was configured with CUDA version 11.7, Tensorflow 2.10.0, and Python 3.10.9.

5.2. Application of FL P2P for DFU Classification

The overall architecture we are proposing for the classification of DFU is based on the Siamese network. The Siamese network was presented in the context of signature verification [18] and comprises two identical networks that take in separate inputs but are connected in the last layer.

Figure 4 gives a high-level view of the Siamese network as a block diagram. The Siamese neural network usually uses contrastive loss [33], which aims to maximize the proximity between positive pairs while simultaneously increasing the dissimilarity between negative pairs.
Figure 4. Block diagram of Siamese network.

For the CNN backbone, we used EfficientNetV2S based on EfficientNet [34] architectures, which have been shown to significantly outperform other networks in classification tasks while having fewer parameters. EfficientNetV2S has fewer parameters, making it more suitable for low-resource settings, and it uses a combination of efficient network design and compound scaling to achieve high accuracy with fewer parameters [35]. The second backbone of the ensemble model is based on Vision Transformers. This was first introduced by the paper “An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale” [36] and is referred to as Vision Transformers (ViTs).

The classification model is a milestone in the development of an innovative tool to be used to assist medical health professionals in performing follow-ups of patient with DFUs. Figure 5 illustrates the proposed approach to DFU classification.

Figure 5. Overall schematics of the innovative framework for DFU classification [17].
The ensemble model architecture we propose to experiment with uses EfficientNetV2S as the CNN backbone and Bidirectional Encoder representation from Image Transformers (BEITs) for the Vision Transformer as proposed by Toofanee et al. [17]. Figure 6 shows the internal architecture of the ensemble model.

**Figure 6.** Block diagram of the ensemble network, illustrating the internal architecture of the individual networks composing the SNN. The CNN utilized is EfficientNetV2S, while the ViT employed is BEiT [17].

In deep learning, the efficacy of a model is highly dependent on the quality and representativeness of its training data. It is imperative to tackle data bias to ensure robust and accurate model performance. This challenge is particularly pronounced in our multi-class classification of diabetes foot ulcers (DFUs), where there may be an uneven representation of classes. As detailed in Section 5.3, the data distribution across classes—both infection, ischemia, and none—is 621, 2555, 227, and 2552, respectively. Notably, the ischemia class is underrepresented with only 227 instances, despite including augmented images.

Constructing a balanced dataset would ideally involve collaboration among various medical centers to share DFU images. While the benefits of artificial intelligence in medical diagnostics are widely recognized, the sharing of medical data raises substantial ethical concerns. To this end, federated learning presents a promising solution. It aims to augment the training dataset while upholding the confidentiality of sensitive medical data by retaining them at the source. This approach not only enhances the model’s training but also addresses privacy and ethical considerations inherent in medical data handling.

### 5.3. Dataset

Data quality is a crucial factor that directly affects the performance of supervised learning algorithms. The utilization of a representative and high-quality dataset is critical for achieving optimal accuracy and performance [37]. In this study, we obtained the dataset from the DFUC2021 challenge organized by the Medical Image Computing and Computer-Assisted Intervention (MICCAI) society [16]. The proper licensing was also secured for this research, ensuring that all ethical and legal requirements were met. Upon initial preprocessing, we observed that the dataset’s class distribution was imbalanced, with 621, 2555, 227, and 2552 instances belonging to the classes both, infection, ischemia, and none, respectively.

### 5.4. Experimental Parameters

We initially aimed to compare the performance of the centralized version of federated learning (FedAVG and FedSGD) with the distributed P2P architecture (FedAVGP2P and
FedSGDP2P). The objective was to use a high number of clients (C = 100, 200, 300, etc.) and a large number of communications (round = 100, 200, 300, etc.) in our experiments to obtain the most relevant results for the purpose of analysis. However, we soon realized that due to resource constraints, the computation times were excessively high, primarily because of the heavy deep learning models used and described earlier, which we also had to substitute for a computation-friendly backbone.

As a result, we decided to limit the maximum number of clients to 20 and the maximum number of rounds to 10. In the case of FedAVG, each round consists of two steps: selecting the clients that receive the aggregated model from the central server, and selecting the clients that send updates of their local models to the central server. In the case of FedAVGP2P and FedSGDP2P, during each communication round, we evaluate the clients' models on the test data. The round concludes when a client receives updates from all its neighbors. The training data are distributed among the clients, considering both IID and non-IID data distribution scenarios.

To evaluate the performance of the three heuristics (n lastest, f1-score, and score cosine), we vary the fraction of clients C with values of 0.1, 0.2, 0.5, and 1.0. As a result, each client communicates with 2, 4, 10, or 20 neighboring clients in each round. After each round, we assess all clients' performance on the test data. During experimentation, the backbones initially proposed could not be used because of resource limitations. We were forced to change the backbones of the ensemble model architecture to a combination of ["MobileNet", "MobileNetV2"]. Table 1 shows the additional parameters used.

<table>
<thead>
<tr>
<th>Table 1. Parameters of model trained.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
</tr>
<tr>
<td>Number of Epochs</td>
</tr>
<tr>
<td>steps per epoch</td>
</tr>
<tr>
<td>Original image Size</td>
</tr>
<tr>
<td>Cropped Image Size</td>
</tr>
<tr>
<td>Activation Function</td>
</tr>
</tbody>
</table>

5.5. Metrics

In our study, the confusion matrix is constructed to evaluate the performance of the multi-class classification model across four distinct classes: both, infection, ischemia, and none. Here is how we define each term within our confusion matrix for each class:

True positives (TPs): These are cases where the model correctly identifies the presence of a condition. For instance, if a case is actually 'both' (meaning it has both infection and ischemia), and our model also predicts 'Both', it is counted as a true positive for that class. Similarly, we count TPs for each of the other classes (infection, ischemia, and none) when the model's prediction matches the actual label.

True negatives (TNs): These are the cases where the model correctly identifies that a condition is not present. For example, for the class 'infection', a true negative is when the model predicts any class other than 'infection' (be it ischemia, both, or none), and the actual class is indeed not 'infection'. This logic applies similarly for the other classes.

False positives (FPs): These occur when the model incorrectly predicts the presence of a condition. Taking 'ischemia' as an example, if the model predicts 'ischemia' when the actual class is ‘none’, ‘infection’, or ‘both’, it would be considered a false positive for 'ischemia'.

False negatives (FNs): These occur when the model incorrectly predicts the absence of a condition. Using the class ‘none’ as an example, a false negative would be when the model predicts either 'infection', 'ischemia', or 'both', but the actual condition is ‘none’.
The output of the confusion matrix is used to calculate the f1-score as shown in Equation (4):

\[
F1-Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times TP}{2 \times TP + FP + FN}
\]  (4)

For multi-class classification with imbalanced data, the main consideration is the Macro f1-score. The formula is illustrated with the following Formula (5), where \( n \) represents the number of class involved. In the DFU classification, \( n \) is equal to 4:

\[
\text{Macro F1 Score} = \frac{\sum_{i=1}^{n} F1 \text{ score}_i}{n}
\]  (5)

5.6. Results

In this section, we present the results obtained and examine the behavior of the centralized model (FedAVG) and various decentralized FedP2P architecture variants, taking into account both IID and non-IID data distributions. An initial observation indicates that the FedP2P architecture, considering all the heuristics, appears to yield stable results compared to those obtained by the centralized FedAVG architecture. A detailed discussion of these results is provided in the discussion section.

In Figure 7a, which considers IID data, it is observed that all algorithms exhibit a linear increase in the number of models sent as the fraction of clients grows. However, the slope of this increase varies among different algorithms, depending on the heuristics used, denoted as H0, H1, H2, and H3. Notably, P2P-H0-AVG and P2P-H0-SGD show a steeper slope, indicating a higher communication cost with increasing client participation. In contrast, algorithms with heuristics H1, H2, and H3 demonstrate more moderate increases, suggesting better communication efficiency.

Figure 7b presents a similar analysis but for non-IID data, which is more representative of real-world scenarios, where data are unevenly distributed across clients. The trends are comparable to those in Figure 7a, with all algorithms experiencing an increase in the number of models sent as more clients participate. Nevertheless, the rate of increase is generally lower for non-IID data, particularly for algorithms utilizing heuristics H1, H2, and H3. This suggests that while the communication overhead is still present, these algorithms are potentially more robust to the challenges posed by non-IID data. In both IID and non-IID scenarios, the SGD variants generally exhibit higher communication costs than their AVG counterparts, which could be attributed to the more frequent model updates required by SGD. The impact of this difference is more pronounced in the IID data scenario.

![Figure 7. Cont.](image-url)
Figure 7. A comparison of FedAVGP2P to FedSGDP2P considering models sent in the network when 90% model accuracy was reached. C is the fraction of clients that the central server (or every client with FedAVGP2P) had received updates from. According to the graph above, it can be said that Heuristic 1 has the best learning ability when we increase factor C.

Figure 8 shows the learning performance of two different centralized federated learning algorithms—FedAVG and FedSGD—under both IID and non-IID data conditions. The graphs show the f1-score as it relates to the number of communication rounds between the central server and clients, with a threshold f1-score of 0.9 marked for reference. The threshold of 0.9 is a significant benchmark, representing a high level of model performance in terms of both precision and recall. Figure 8b shows the performance of the centralized FedAVG with non-IID data. The threshold is reached when communication round = 4. It should be noted that at communication round = 8, centralized FedAVG with non-IID data is still learning.

Figure 8. Execution of centralized FedAVG and FedSGD with f1-score threshold set at 0.9.
Figure 8c presents the centralized FedSGD algorithm on IID data. The f1-score threshold is crossed when communication rounds equal 4. Figure 8d displays the centralized FedSGD algorithm on non-IID data. The model reaches the f1-score threshold when the communication round is 4, and it peaks when the communication round is 7. This implies that FedSGD struggles with learning from non-IID data and requires additional rounds.

Figure 9 shows the performance of peer-to-peer federated learning algorithms, specifically FedAVG and FedSGD, with an f1-score threshold set at 0.9 and Heuristic 0 when the fraction of clients is varied. Figure 9a,b show the results from the execution of FedAVG-P2P on IID and non-IID data, respectively. Across both data types, the threshold is crossed at the same time, despite varying the number of clients.

Figure 9c,d show the execution for FedSGD-P2P. The threshold for the f1-score is reached at communication round 4 for both data types.

Overall, the performance of both FedAVG-P2P and FedSGD-P2P with Heuristic 0 and a varying fraction of clients indicates that while the fraction of client adjustments might slightly affect the speed of convergence to the threshold f1-score, the algorithms are generally capable of reaching the desired performance level regardless of the C setting, especially in IID data scenarios. In non-IID scenarios, where client data are more heterogeneous, the choice of C appears to be more critical.

Figure 10a shows that the performance of the model performs consistently across different values of fraction of clients C when the data are IID. The convergence towards the f1-score threshold is smooth, indicating that the heuristic and IID assumptions work well together. However, there is a notable performance variance with different ‘C’ values, suggesting that ‘C’ impacts the learning process when data are uniformly distributed. From Figure 10b, we can see that the threshold is crossed when the round is 4, just as...
Figure 10c is executed under IID conditions and shows a slightly erratic convergence, potentially indicating a greater sensitivity to the stochastic nature of the algorithm. The choice of C affects the rate of convergence, implying its role as a tuning parameter for balancing communication efficiency and model performance. Finally, Figure 10d reveals a more pronounced effect of C values on performance. It also shows that the threshold is reached quicker at approximately the communication round equal to 3.

Figure 10. Execution of FedAVG2P and FedSGD2P with f1-score threshold set at 0.9 and Heuristic 1.

Figure 11 explores the performance when Heuristic H2 is applied to the FedAVG2P and FedSGD2P algorithms across different fractions of clients (denoted as ‘C’) in both IID and non-IID settings. Figure 11a suggests that when data are identically and independently distributed, the FedAVG2P algorithm exhibits stable performance across varying fractions of client participation. Interestingly, the performance difference between the various C values is marginal, indicating that Heuristic H2 may enable the algorithm to leverage information effectively, even with lower client participation. In Figure 11b, in contrast to IID data, the non-IID setting reveals a wider spread in performance across different C values. Both cross the threshold at approximately when the communication round is 4. Figure 11c shows the performance pattern for FedSGD2P under IID conditions with Heuristic H2. It tends to mirror that of FedAVG2P, with a notable difference in threshold crossing at communication round at 3. The convergence of the f1-score towards the threshold, irrespective of the C value, points to a potential reduction in the necessity for high client participation. The non-IID scenario for FedSGD2P is illustrated in Figure 11d, where the threshold f1-score is crossed when the communication round is approximately 4.
Figure 11. Execution of P2P FedAVG and FedSGD with f1-score threshold set at 0.9 and Heuristic 2.

Figure 12a illustrates that the FedAVGP2P algorithm under IID conditions with Heuristic H3 shows a close convergence of f1-scores for all C values by the eighth communication round. The threshold of the f1-score is crossed when the communication round is approximately 4. The same applies for Figure 12b with the difference that the threshold is crossed when the communication round is approximately 3. Figure 12c shows that the threshold is approximately crossed when the communication round is 5. The convergence of performance across different fractions of client participation indicates that Heuristic H3 aids in efficient learning, irrespective of the exact participation rate. In Figure 12d, the graph reveals a greater spread in performance between different ‘C’ values, especially in the initial rounds. However, as the communication rounds increase, the performance for all ‘C’ values tends to converge. The threshold is crossed when the communication round is approximately 3.

Figure 12. Cont.
Table 2 provides a summary of the communication efficiency of various federated learning algorithms, comparing how many communication rounds are needed to cross a predefined f1-score threshold, which is set at 0.9.

Table 2. Summary of communication rounds needed to cross f1-score threshold.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Communication Rounds to Cross f1-Score Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized FedAVG IID</td>
<td>3</td>
</tr>
<tr>
<td>Centralized FedAVG non-IID</td>
<td>4</td>
</tr>
<tr>
<td>Centralized FedSGD IID</td>
<td>4</td>
</tr>
<tr>
<td>Centralized FedSGD non-IID</td>
<td>4</td>
</tr>
<tr>
<td>FedAVGP2P Heuristic 0, IID</td>
<td>approx 4</td>
</tr>
<tr>
<td>FedAVGP2P Heuristic 0, non-IID</td>
<td>approx 4</td>
</tr>
<tr>
<td>FedSGDP2P Heuristic 0, IID</td>
<td>4</td>
</tr>
<tr>
<td>FedSGDP2P Heuristic 0 non-IID</td>
<td>3</td>
</tr>
<tr>
<td>FedAVGP2P Heuristic H1, IID</td>
<td>4</td>
</tr>
<tr>
<td>FedAVGP2P Heuristic H1, non-IID</td>
<td>4</td>
</tr>
<tr>
<td>FedSGDP2P Heuristic H1, IID</td>
<td>4</td>
</tr>
<tr>
<td>FedSGDP2P Heuristic H1, non-IID</td>
<td>approx 3</td>
</tr>
<tr>
<td>FedAVGP2P Heuristic H2, IID</td>
<td>approx 3</td>
</tr>
<tr>
<td>FedAVGP2P Heuristic H2, non-IID</td>
<td>4</td>
</tr>
<tr>
<td>FedSGDP2P Heuristic H2, IID</td>
<td>approx 3</td>
</tr>
<tr>
<td>FedSGDP2P Heuristic H2, non-IID</td>
<td>4</td>
</tr>
<tr>
<td>FedAVGP2P Heuristic H3, IID</td>
<td>4</td>
</tr>
<tr>
<td>FedAVGP2P Heuristic H3, non-IID</td>
<td>approx 4</td>
</tr>
<tr>
<td>FedSGDP2P Heuristic H3, IID</td>
<td>5</td>
</tr>
<tr>
<td>FedSGDP2P Heuristic H3, non-IID</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 13 illustrates the f1-score trajectories of an SGD-based learning algorithm under IID and non-IID conditions, comparing the scenarios with and without the application of self-training.
5.7. Discussions

In this section, we discuss and analyze the results obtained. Considering the convergence behaviors of the models, the results indicate that the models trained with FedAVG and FedSGDP2P can achieve comparable behaviors to FedAvg when provided with both IID and non-IID client data.

From Table 2, some important points come to light. It shows that, generally, the number of communication rounds needed to cross the f1-score threshold varies depending on whether the data are IID or non-IID, with non-IID data often requiring more rounds. It further indicates that algorithms tend to reach the f1-score threshold more quickly with IID data than with non-IID data. This is expected, as non-IID data represent a more realistic but challenging scenario, where data are unevenly distributed across clients, which can complicate the learning process. Centralized algorithms, both FedAVG and FedSGD, show a consistent requirement for communication rounds, irrespective of the data distribution (IID or non-IID). In contrast, heuristic optimizations in decentralized settings (P2P) display a variation in the number of rounds needed, which could indicate that certain heuristics are better suited for specific data distributions.

By observing the convergence behaviors of the models for FedAVG and FedP2P, we observe that the general behaviors are quite similar for both methods. Most experiments conclude with models reaching an accuracy of approximately 92%. These results suggest that the convergence behaviors of the average FedAVGP2P models are more comparable to those of FedAVG when the size of C is sufficient.

Let us consider the experiments with the fewest models sent over the network when the model f1-score reached 90%: in both cases, with IID and non-IID client data, both FedAVG and FedSGDP2P required higher network communication costs (number of rounds). However, naturally, with FedAVGP2P, the burden of communication costs is distributed among the participating clients rather than being heavily concentrated on a central server. Therefore, if there is a communication constraint at the central server level, such as insufficient bandwidth, FedAVGP2P may be a suitable approach.

Regarding the effect of the heuristics, for higher values of C, we observe comparable convergence behaviors for all the algorithms. This partly indicates that when communicating with a large portion of clients in the network, the choice of neighbors with whom each client communicates is not of significant importance. This situation makes us push our analysis further as to why the use of heuristics did not perform better than the original FedAVGP2P and FedSGDP2P. One possible reason could be that the heuristic leads the network clients to communicate more frequently with the same type of neighbors. This, in turn, could introduce multiple clusters in the network, where clients are more likely to communicate with neighbors within the same cluster. Additionally, this could prolong the time.
during which clients receive model parameters from neighbors outside their own cluster, potentially leading to lower performance by reducing the diversity of model parameters received by each client.

5.8. Limitations

A limitation of the study is the lack of resources to train richer models on our dataset of DFU with our high-performing deep learning Siamese model with a CNN and ViT backbones. We were also limited in terms of the number of epochs and number of clients.

6. Conclusions

The overall results presented in this article indicate that training a model using a P2P FL architecture could be a viable approach for collaborative neural network modeling among multiple clients without sharing their training data. Firstly, the results show that models trained with FedAVGP2P and FedSGDP2P are comparable to models trained with the centralized FedAVG architecture in terms of accuracy. FedP2P may be less desirable due to higher global network costs compared to FedAVG, as more data need to be transmitted to achieve comparable model convergence behaviors. However, the use of a P2P topology offers several advantages, such as the absence of a single point of failure and dependence on a central server. This makes P2P FL a wise choice if these characteristics are required.

As future work, it would be interesting to investigate whether these clusters emerge by analyzing the choice of neighbors for each client throughout the training process. It would also be valuable to explore the scenarios in which FedAVGP2P or FedSGDP2P would be faster than FedAVG, taking into account the training time. The answer to this question depends on various factors, such as communication constraints and client systems.

For instance, using FedAVG could be a faster approach if the central server has sufficient bandwidth. However, FedAVGP2P could also be faster if the central server lacks such bandwidth. Looking at certain curves related to the heuristics of the FedAVGP2P and FedSGDP2P algorithms, we observe the influence of the f1-score achieved based on the number of rounds and the fraction of clients. This indicates the possibility of studying the trade-off between precision and communication according to the methods used.

Furthermore, at a fixed precision level, the different methods yield varying numbers of rounds, which can be utilized to measure the communication cost of each method. Similarly, we can explore and compare the methods to find the one that achieves the best precision at a fixed communication cost.

To further refine the relevance of our results, additional measurements should be implemented by increasing the number of collaborative clients. Our results in P2P FL, through FedAVGP2P and FedSGDP2P, demonstrate it as a promising approach for training neural network models across multiple clients. The experiments conducted in this paper and the subsequent results clearly show that there are options for ensuring the confidentiality of data in a medical setup, where massive and sensitive data are needed to have an optimized model. Security features, in terms of privacy, can further be added by exploring the possibilities offered by homomorphic encryption.

Author Contributions: Conceptualization, M.S.A.T., M.H., S.D. and K.T.; Methodology, S.D. and K.T.; Software, V.P. and A.K.D.; Validation, M.S.A.T. and M.H.; Formal analysis, M.S.A.T.; Resources, V.P.; Writing—original draft, M.S.A.T.; Writing—review & editing, M.S.A.T., M.H., S.D. and K.T.; Supervision, D.S. All authors have read and agreed to the published version of the manuscript.

Funding: The Research was funded by XLIM, UMR CNRS 7252, University of Limoges.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.
Acknowledgments: We would like to thank the research team “U1094 Inserm U270 EpiMaCT Épidémiologie des maladies chroniques en zone tropicale” of University of Limoges, which has contributed to this work by offering the processing power of their server for the GPU processing.

Conflicts of Interest: The authors declare no conflict of interest.

References


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.