MNFL: A METHOD TO DEAL WITH DATA IMBALANCE WITH DISTRIBUTED MEDICAL IMAGING DATASETS

A PREPRINT

Erfan Darzidehkalani[1,2] **P.M.A. van Ooijen**[1,2]

[1]Department of Radiation Oncology, University Medical Center Groningen,
University of Groningen, Hanzeplein 1, Groningen, the Netherlands

[2]Machine Learning Lab, Data Science Center in Health (DASH),University Medical Center Groningen,
University of Groningen, Hanzeplein 1, the Netherlands

{e.darzidehkalani,p.m.a.van.ooijen}@umcg.nl

November 14, 2021

ABSTRACT

Annotating data is expensive and time consuming, and it is difficult for a single medical center to achieve a sample size large enough to create its own personalized model. Alternatively, you can pool data from all centers to train a centralized model that anyone can use. However, because medical data is sensitive to data protection, such strategies are often not feasible. Federated learning (FL) was recently introduced to collaborate on common predictive models between centers without exchanging data. FL was vastly used in COVID-19 research, where multiple centers collaborated on finding decentralized solutions to process COVID-19 imaging data. This article introduces MNFL, a new FL framework designed specifically for medical applications. However, all of them rely on model parameters aggregation at server side that employ same model structure and size. Solutions for collaboratively training client side models for multiple clients are missing from status quo, or difficult for deployment in practical scenarios. Inspired by the concept of knowledge distillation (KD) and its capability of transferring knowledge from a neural network to another via exchanging the logits output, he standard formulation of FL produces one shared model for all clients which ignores the personalization of each client, especially given the heterogeneity of data distribution for various clients.However, this architecture-agnostic training framework only develops a global output(/logits) for all the clients, and therefore, it does not adapt the personalization to each client during aggregation, especially for clients with heterogeneous data distributions.Although there are already plenty of work in efforts to achieve personlization in FL (e.g., regularization-based method), most of them rely on the model parameters in the server, which is impractical in FD. This paper is an attempt to deal with the aformentioned issue with an effectice collaortive algorithm focusing on client side of the network. And works superior to the state-of-the-art algorithms in COVID-19 detection task.

Keywords Federated learning · Privacy-preserving machine learning · Medical imaging

1 Introduction

Coronavirus first emerged in Wuhan, China, in 2019, [1]. Coronavirus is known as viral pneumonia, and this viral pneumonia can be grouped into COVID-19, SARS, and MERS. A significant outbreak of CoronaVirus first happened in China. Corona is a global problem right now which affects many aspects of human life. It can be spread by human-human interaction. There are several ways to diagnose COVID-19. Transcription-polymerase chain reaction

(RT-PCR) tests are one of the most common ways of diagnosing COVID-19. However, there are other ways of diagnosing COVID-19 [2].

Research has shown the effectiveness of chest imaging in the diagnosis of COVID-19 infected people. Chest imaging can be used as a powerful tool to diagnose COVID-19. However, due to the large number of people having the symptoms and being tested for COVID-19, it might not always be feasible for radiologists to examine a large number of patients during the outbreak. Scans of CT and X-ray of infected patients have deformation, misalignment between certain parts, or pixel intensities different from ordinary people in infected parts. Observing this misalignment is not an easy task and requires effort and focus from radiologists to correctly distinguish between healthy and infected people. Instead, machine-aided diagnoses, such as deep learning models, can help radiologists to detect COVID-infected patients and save much time for radiologists.

Deep learning methods, such as Convolutional neural networks (CNN), has shown great promise in finding infected areas in images. They have shown to be working on CT-SCans, and X-ray images of COvid infected people. Various architectures of CNN have been tested and shown outstanding performance on the imaging datasets. Models such as AlexNet [3], ResNet[4] and MobileNet [5] could classify thousands of objects belonging to hundreds of classes with the same accuracy as humans. The deep learning models have also been used in medical imaging, with excellent performance. Deep learning models have been used to classify cancer lesions [6] brain disease classification [7],retinal image segmentation[8], lung segmentation [9], diagnosis of breast cancer[10]. It has also been used to detect COVID-19 from imaging data. Most of the use cases were either for x-rays like the researches in X-Ray or CT images. While CT scans can help screen COVID-19 potential infected people, CT scans of other diseases could also have overlapping features with COVID injected patients. Hence, it is tricky to distinguish COVID infected patients from other types of lung inflammation and requires radiologist-level expertise. Most literature focuses on the ways to distinguish better COVID infected patients from other diseases.

The data used in these studies could generally be categorized in two ways. In some studies, they use local data with a small sample size. The performance of the models developed on local data is generally good. However, the generalizability of such models trained on limited data is always under question. Considering the difference and variety in data samples in different hospitals, a deep learning model trained is biased towards the local data it was exposed and could not be a reliable model in other domains., This type of training lacks collaboration between different hospitals.

Another type of training is accumulating various datasets from different hospitals. Using all of the data in a central data station training a deep learning model based on that data. This way of training can address the generalizability problem and makes use of a large, diverse dataset. However, there is no actual collaboration among centers, they instead collected data from various sources. A barrier in collecting data from various sources and practices is that medical data could not be accumulated considering the strict privacy laws. GDPR[4], for example, is designed to safeguard users' personal privacy and data security by taking the General Data Protection Regulations, introduced by the European Union on 25 May 2018. Plus, the researches done in this way use the confidential data accessible by the institutes participating in that research and not accessible by third-party users[11], [12] , [13], [14], [15] [16].

One potential solution to this problem is federated deep learning. Federated learning aims to decentralize the whole process of training and training the model while leaving the data in the sites. In this paper, we develop a framework that enables collaboration between hospitals and uses multiple data sources to detect COVID-19 infection using federated learning. The decentralized way of distributing data among different centers guarantees privacy. Besides, the distributed computation and development of deep learning models ensure the accumulation of knowledge and reaching the full capacity of deep learning models. There are several ways of doing federated learning since federated learning is performed at a higher level compared to conventional deep learning. There are more parameters of complexity since the model should take into consideration the general and local methods of optimization, data accumulation for all the sites, and local training models. There are several parameters playing a role in the final performance of a federated learning setting.

2 Related works

2.1 Covid detection

Several recent studies have been done in order to classify the COVID-19 images from normal images and to locate the lesion areas. A three-dimensional deep neural network was implemented by Wang by integrating activated regions with unattended connected components in the classification network, attaining 90,1 percent accuracy and 95,5 percent ROC. This 3D classification model does, however, have drawbacks, such as a long training period. Zhao

[17] has proposed a data-instance prediction mechanism-based pooling process that achieves 97.9% accuracy and 99% AUC in the COVID-19, CP, and normal classification tasks, although there are restricted test set size issues, For example, 3D-based prediction. Horry [18] has suggested a COVID-19 detection process that builds on migration and multi-modal imaging data. However, the classification model created is not performant enough. For COVID-19 and non-COVID-19 classifications of 98.37 percent accuracy and AUC 98.32 percent, Pathak et al. [19] proposed a deep bidirectional long-short-speed memory network, using a DBM model for COVID-19. Wang et al. [20] have developed a COVID-19 technique for deep learning on X-rays and model integration, with strong results, 96.1 percent in predictive tri-classification tasks, although only 140 pictures from COVID-19 exist.

2.2 Federated Learning

FL is a learning paradigm where several people work together without the requirement to exchange or centralize data sets. In practice, each participant generally gets and refines a global consensus model by doing local optimization rounds, possibly directly or via a parameter server, before exchanging changes. The more local training rounds are carried out, the less the whole method is certain to find a global equilibrium. Aggregation methods might be based on a single trusted server and multiple site models or several nodes without a central system. By letting different parties work together without the requirement to exchange or centralize data, FL deals with problems relating to the absence of sensitive health data. As a result, new research and commercial options may be opened up, and patient care can be improved on an international scale if done properly. Today, however, FL has an effect on almost all actors and the whole therapy cycle. Collaborative learning has been embraced by more and more academics and companies nowadays when it stresses data privacy and data security.

3 Materials and Methods

In the real world setting, institutions are not willing to share their confidential data. In reality, hospitals and other relevant organizations are reluctant to share their patients' data to preserve the privacy of the patients. Moreover, it is a known fact that deep learning models require a large amount of data to train a model that can handle real-world problems. For that reason, this paper considers collecting multiple hospitals' data without leakage of data privacy. The federated learning algorithm was implemented in order to compare and find the results from various methods and in different settings.

3.1 Problem Formulation

We aim to collaboratively train client side models for a set of clients applying different model structures in FL. Consider supervised learning whose goal is to learn a function that maps every input data to the correct class out of $\mathcal{C}$ possible options. We assume that there are N clients, and each client n can only access to his private dataset $\mathbb{D}_n := \{x_i^n, y_i\}$, where x_i is the i-th input data sample, y_i is the corresponding label of x_i, $y_i \in \{1, 2, \cdots, C\}$. The number of data samples in dataset $\mathbb{D}_n$ is denoted by D_n. $\mathbb{D} = \{\mathbb{D}_1, \mathbb{D}_2, \cdots, \mathbb{D}_N\}$, $D = \sum_{n=1}^{N} D_n$. In conventional FL, the goal of the learning system is to learn a global model $\mathbf{w}$ that minimizes the total empirical loss over the entire dataset $\mathbb{D}$:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := \sum_{n=1}^{N} \frac{D_n}{D} \mathcal{L}_n(\mathbf{w}), \text{ where } \mathcal{L}_n(\mathbf{w}) = \frac{1}{D_n} \sum_{i=1}^{D_n} \mathcal{L}_{CE}(\mathbf{w}; x_i, y_i), \tag{1}$$

where $\mathcal{L}_n(\mathbf{w})$ is the n-th client's local loss function that measures the local empirical risk over the private dataset $\mathbb{D}_n$ and $\mathcal{L}_{CE}$ is the cross-entropy loss function that measures the difference between the predicted values and the ground truth labels.

However, this formulation requires all local clients to have a unified model structure, which cannot be extended to more general cases where each client applies a unique model. Therefore, we need to reformulate the above optimization problem to break the barrier of homogeneous model structure.

$$\min_{\mathbf{w}=[\mathbf{w}^1, \cdots, \mathbf{w}^N] \in \mathbb{R}^{\sum_{n=1}^{N} d_n}} L(\mathbf{w}) := \sum_{n=1}^{N} \frac{D_n}{D} \mathcal{L}_n(\mathbf{w}^n) \tag{2}$$

Moreover, clients' data may come from isolated environments, e.g., different contexts and applications, the minimization of the overall local loss might not be the ideal solution for a given client given that its data distribution differs from the global distribution significantly. Besides, given the Non-IID clients' datasets, it is inappropriate to just minimize the total empirical loss (i.e., $\min_{\mathbf{w}^1, \cdots, \mathbf{w}^N} \mathcal{L}(\mathbf{w}^1, \cdots, \mathbf{w}^N) := \sum_{n=1}^{N} \frac{D_n}{D} L_n(\mathbf{w}^n)$).

To that end, we propose the following training framework: Let $s(\mathbf{w}^n, \hat{x})$ denote the *collaborative knowledge* from client n, and $\hat{x}$ denote a data sample from a public dataset $\mathbb{D}_r$ that all clients can access to. Define the client side loss function of client n as

$$\mathcal{L}_{per,n}(\mathbf{w}^n) := \mathcal{L}_n(\mathbf{w}^n) + \lambda \sum_{\hat{x}\in\mathbb{D}_r} \mathcal{L}_{KL}\left(\sum_{m=1}^{N} c_{mn} \cdot s(\mathbf{w}^m, \hat{x}), s(\mathbf{w}^n, \hat{x})\right), \tag{3}$$

where $\lambda > 0$ is a hyper-parameters, $\mathcal{L}_{KL}$ stands for Kullback–Leibler (KL) Divergence function and is added to the loss function to transfer client side knowledge from a teacher to another. c_{mn} is the knowledge coefficient which is used to estimate the contribution from client m to n. The second term in overall loss allows each client to build his own client side aggregated knowledge in the server and enhance the collaboration effect between clients with large $\mathbf{c}$. The concept of *collaborative knowledge* can either refer to soft predictions or model parameters depending on the definition in different situations. For example, $s(\mathbf{w}^n, \hat{x})$ can be deemed to be a soft prediction of the client n, which are calculated with the softmax of logits z^n, i.e., $s(\mathbf{w}^n, \hat{x}) = \frac{\exp(z_c^n/T)}{\sum_{c=1}^{C}\exp(z_c^n/T)}$, logits z^n is the output of the last fully connected layer on client n's model, T is the temperature hyperparameter of the softmax function.

We define $\mathbf{c} \in \mathbb{R}^{N\times N}$ as the *knowledge coefficient matrix*:

$$\mathbf{c} = \begin{Bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \cdots & c_{NN} \end{Bmatrix}. \tag{4}$$

Our objective is to minimize

$$\min_{\mathbf{w},\mathbf{c}} \mathcal{L}(\mathbf{w}, \mathbf{c}) := \sum_{n=1}^{N} \frac{D_n}{D} \mathcal{L}_{per,n}(\mathbf{w}^n) + \rho\|\mathbf{c} - \frac{\mathbf{1}}{N}\|^2, \tag{5}$$

where $\mathbf{w} = [\mathbf{w}^1, \cdots, \mathbf{w}^N] \in \mathbb{R}^{\sum_{n=1}^{N} d_n}$ is the concatenated vector of all weights, d_n represents the dimensions of model parameter $\mathbf{w}^n$. $\mathbf{1} \in \mathbb{R}^{n^2}$ is the identity matrix whose elements are all equal to 1. The second term in overall loss is a regularization term that ensures generalization ability of the whole learning system. Without the regularization term, a client with a completely different data distribution tends to set large values of the knowledge coefficient (i.e., equal to 1), and no collaboration will be conducted during training in this case. ρ is a regularization parameter that is larger than 0.

3.2 Distributed solutions

All of the implemented federated learning models are centralized models, which means that there is a central trusted node, which controls the whole process of training. In the centralized paradigm of federated learning, multiple sites exchange updates of their models trained on their local data without releasing the actual data they are trained on. Each site has a two-way exchange with the central server, i.e., it both takes an updated model from the central server and also sends a model trained on its private dataset. In the local training phase, a few epochs of local optimization of the model are performed. Then, the updates are shared through the local server. The way that the sites send their local updates and the central server aggregates the data defines the federated learning topology. In the real world setting, the actual federated learning model is dependent on the network design, desired model performance, and implementation limitations like bandwidth issues and legal constraints. Although there is a debate on core concepts of federated learning, such as system constraints, evaluation metrics, and optimization methods, in this paper, we tried to cover the most important ones which are widely used and in their broadest definition. In the following, you can find the methods that we used and examined in local settings.

3.3 MNFL Algorithm

In this section, we introduce the proposed MNFL algorithm, where the local model parameters and knowledge coefficient matrix are updated alternatively. To enable client side knowledge transfer in FL, we train client side models locally according to the related *collaborative knowledge*. We can design an alternating optimization approach to solve the optimization problem. that in each round we fix either $\mathbf{w}$ or $\mathbf{c}$ by turns, and optimize the unfixed one following an alternating way until a convergence point is reached.

Update $\mathbf{w}$: In each communication round, we first fix $\mathbf{c}$ and optimize (train) $\mathbf{w}$ for several epochs locally. In this case, updating $\mathbf{w}$ depends on both the private data (i.e., $\mathcal{L}_{CE}$ on $\mathbb{D}_n, n \in [1, \cdots, N]$), that can only be accessed by the

corresponding client, and the public data (i.e., $\mathcal{L}_{KL}$ on $\mathbb{D}_r$), which is accessible for all clients. We propose a two-stage updating framework for $\mathbf{w}$:

- *Local Training*: Train $\mathbf{w}$ on each client's private data by applying a gradient descent step:

$$\mathbf{w}^n \leftarrow \mathbf{w}^n - \eta_1 \nabla_{\mathbf{w}^n} \mathcal{L}_n(\mathbf{w}^n; \xi_n), \tag{6}$$

  where ξ_n denotes the mini-batch of data $\mathbb{D}_n$ used in local training, η_1 is the learning rate.
- *Distillation*: Transfer knowledge from client side soft prediction to each local client based on public dataset:

$$\mathbf{w}^n \leftarrow \mathbf{w}^n - \eta_2 \nabla_{\mathbf{w}^n} \mathcal{L}_{KL}\left(\sum_{m=1}^{N} \mathbf{c}_m^{*,T} \cdot s(\mathbf{w}^m, \xi_r), s(\mathbf{w}^n, \xi_r)\right), \tag{7}$$

  where ξ_r denotes the mini-batch of public data $\mathbb{D}_r$, and η_2 is the learning rate. $\mathbf{c}_m^* = [c_{m1}, c_{m2}, \cdots, c_{mN}]$ is the *knowledge coefficient* vector for client m, which can be found in m-th row of $\mathbf{c}$. Note that all *collaborative knowledge* and *knowledge coefficient matrix* are required to obtain the client side soft prediction in this stage, which can be collected in the server.

Federated Averaging In the federated averaging, a model tries to minimize a global loss function, which is a sum of local loss functions. Each client only has access to its own data and a respective loss function for training on that data. Suppose there are K clients participating in the federated network. There is a model proposed by [21], called the model averaging method. In this method, assuming in round t, the global model is updated by the updates from K-th client. The gradient of the K-th client can be seen in Eq. '1'.

$$g_k = \nabla F_k(w_t) \tag{8}$$

$$\omega_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k, \tag{9}$$

global model parameters are shown as ω_t, and local gradient is shown as g_k. η is the learning rate. Global model is updated according to the clients' gradients using the formulation update Eq. '1.5'. The global model gradient is average of local gradients.

$$\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t) \tag{10}$$

An equivalent method, averaging of federated models, is also proposed by the authors. In this method, local clients train models on their own data, and then a central server aggregates the trained models. The local models are updated according to Eq. '2', and the global server updates the global model by averaging the local models according to Eq. '3'

$$\forall k, \omega_{t+1}^{(k)} \leftarrow \overline{\omega}_t - \eta g_k, \tag{11}$$

$$\overline{\omega}_{t+1} \leftarrow \sum_{k=1}^{K} \omega_{t+1}^{(k)} \tag{12}$$

Update c: After updating $\mathbf{w}$ locally for several epochs, we turn to fix $\mathbf{w}$ and update $\mathbf{c}$ in the server.

$$\mathbf{c} \leftarrow \mathbf{c} - \eta_3 \lambda \sum_{n=1}^{N} \frac{D_n}{D} \nabla_{\mathbf{c}} \mathcal{L}_{KL}\left(\sum_{m=1}^{N} \mathbf{c}_m \cdot s(\mathbf{w}^{m,*}, \xi_r), s(\mathbf{w}^{n,*}, \xi_r)\right) - 2\eta_3 \rho(\mathbf{c} - \frac{\mathbf{1}}{N}), \tag{13}$$

where η_3 is the learning rate for updating $\mathbf{c}$.

Algorithm alg:1 demonstrates the proposed MNFL algorithm and the idea behind it is shown in Performance Guarantee. Theorem theorem:1 provides the performance analysis of the client side model when each client owns Non-IID data. Detailed description and derivations are deferred to Appendix A. Denote the n-th local distribution and its empirical distribution by $\mathcal{D}_n$ and $\hat{\mathcal{D}}_n$ respectively, and the hypothesis $h \in \mathcal{H}$ trained on $\hat{\mathcal{D}}_n$ by $h_{\hat{\mathcal{D}}_n}$. There always exist $c_{m,n}^*, m = 1, \ldots, N$, such that the expected loss of the client side ensemble model for the data distribution $\mathcal{D}_n$ of client n is not larger than that of the single model only trained with local data: $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^{N} c_{m,n}^* h_{\hat{\mathcal{D}}_m}) \leq \mathcal{L}_{\mathcal{D}_n}(h_{\hat{\mathcal{D}}_n})$. Besides, there

Algorithm 1 MNFL Algorithm

Input: $\mathbb{D}$, $\mathbb{D}_r$, η_1, η_2, η_3 and T
Output: $\mathbf{w} = [\mathbf{w}^1, \cdots, \mathbf{w}^N]$
Initialize $\mathbf{w}_0$ and $\mathbf{c}_0$
procedure SERVER-SIDE OPTIMIZATION
  Distribute $\mathbf{w}_0$ and $\mathbf{c}_0$ to each client
  for each communication round $t \in \{1, 2, ..., T\}$ **do**
    for each client n **in parallel do**
      $\mathbf{w}_{t+1}^n \leftarrow ClientLocalUpdate(n, \mathbf{w}_t^n, \mathbf{c}_{t,n})$
    end for
    Update knowledge coefficient matrix $\mathbf{c}$ via (eq:c)
    Distribute $\mathbf{c}_{t+1}$ to all clients
  end for
end procedure
procedure CLIENTLOCALUPDATE($n, \mathbf{w}_t^n, \mathbf{c}_{t,n}$)
  Client n receives $\mathbf{w}_t^n$ and $\mathbf{c}_n$ from the server
  for each local epoch i from 1 to E **do**
    for mini-batch $\xi_t \subseteq \mathbb{D}_n$ **do**
      Local Training: update model parameters on private data via (eq:w1)
    end for
  end for
  for each distillation step j from 1 to R **do**
    for mini-batch $\xi_{r,t} \subseteq \mathbb{D}_r$ **do**
      Distillation: update model parameters on public data via (eq:w2)
    end for
  end forreturn local parameters $\mathbf{w}_{t+1}^n$
end procedure=0

exist some problems where the client side ensemble model is strictly better, i.e., $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^N c_{m,n}^* h_{\hat{\mathcal{D}}_m}) < \mathcal{L}_{\mathcal{D}_n}(h_{\hat{\mathcal{D}}_n})$.

Theorem theorem:1 indicates that the performance of the client side ensemble model under some suitable coefficient matrices is better than that of the model only trained on its local private data, which theoretically demonstrates the necessity of the parameterized personalization. However, it is challenging to find such matrices due to the complexity and diversity of machine learning models and data distributions. In this paper, our designed algorithm MNFL can find the desired coefficient matrix in a gradient descent manner, hence achieving the performance boost. Besides, we have the similar claim for the relationship between the client side ensemble model and average ensemble model.

There always exist $c_{m,n}^*, m = 1, \ldots, N$, such that the expected loss of the client side ensemble model for the data distribution $\mathcal{D}_n$ of client n is not larger than that of the average ensemble model: $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^N c_{m,n}^* h_{\hat{\mathcal{D}}_m}) \leq \mathcal{L}_{\mathcal{D}_n}(\frac{1}{N}\sum_{m=1}^N h_{\hat{\mathcal{D}}_m})$. Besides, there exist some problems where the client side ensemble model is strictly better, i.e., $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^N c_{m,n}^* h_{\hat{\mathcal{D}}_m}) < \mathcal{L}_{\mathcal{D}_n}(\frac{1}{N}\sum_{m=1}^N h_{\hat{\mathcal{D}}_m})$.

Centralized Data Sharing In the centralized data sharing setting, all of the data are gathered in one data center, which means that the training will be done in a normal, non-distributed manner. This way of distributing data could present a ground truth for comparison between different federated settings and also non-federated settings. One thing to take into consideration is that centralized data sharing has its own hyperparameters, which are general hyperparameters for deep neural networks, like batch size, momentum, and learning rate. But other federated learning algorithms require a method with their own hyperparameters for training, separate model update, and the method to aggregate models require more training parameters. Thus, there

Single Weight Transfer In this method, the model will meet each institution only once. Every institution will train the model for a determined number of epochs and then will pass it to the next institution. Cyclic weight transfer In the cyclic weight transfer model, the model will meet each institution more than once. The loop continues for a determined number of cycles. Also, each institution will have a determined number of epochs. Cyclic weight transfer will require more time than single weight transfer to train. And the convergence might be a bit more tricky since the feature space is high-dimensional and a more general training method. Some algorithms tried to address these challenges by optimizing the hyper-parameters and making them robust or adaptive.

Federated Stochastic Gradient Descent From the total number of clients, a subset is selected, the ratio of selected clients to the total number of clients is defined by C. In federated stochastic gradient descent $C < 1$, for $C = 1$, the training would be non-stochastic (full batch) since all the clients are involved. The gradient will be computed over the selected batch of clients. With a fixed learning rate η for each client, k will do the local learning.

Local models could also have their own epochs, i.e., each local dataset is passed multiple times through the model that, and the updated model is sent to the server after multiple local updates. $w^k \leftarrow w^k - \eta \nabla F_k(w^k)$ multiple times before its sent back to the central server. If $C = 1$, This will be federated averaging (or FedAvg). Three parameters determine the computation load, the number of clients involved in each round of federated training which could be addressed by the fraction parameter C, Number of Local epochs E, and batch size for each client. for $E = 1$ and $C < 1$ the algorithm will be FedSGD.

4 Experiments

In this section, we discuss the experiment and how the COVID-19 datasets were assigned to local clients.

Algorithm 2 Federated Averaging: K is total number of clients, M is clients batch size, each client's internal epoch is shown as E and learning rate is declared as η

Server training
  Initialize ω_0
  for Each round t **do**
    for Each client k **do**
      $\omega_{t+1}^{(k)} \leftarrow$ local training$(k, \overline{\omega_t})$
    end for
    $\overline{\omega}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^{(k)}$
  end for=0

Local training $(k, \overline{\omega}_t)$
  for Each internal epoch i from 1 to S **do**
    $D_m \leftarrow$ Split local training set into multiple batches
    for each batch from 1 to $B = \frac{n_k}{M}$ **do**
      $\omega_{b+1}^{(k)} \leftarrow \omega_{b,i}^{(k)} - \eta \nabla F_k(w_t)$
    end for
  end for
  return $\omega_{t+1}^{(k)} = \omega_{B,S}^{(k)}$ to server =0

4.1 Dataset

Two distinct dataset sources were used to ensure data heterogeneity in the classification task. The first source was the publicly available Tongji hospital Covid CT scans. It consisted of 349 CT scans of CT image slices, as well as 397 Non-Covid patients. The images were low-quality chest CT images. In addition, the SARS-CoV-2 CT scan dataset, consisting of 1252 CT scans, positive for COVID data was used, another 1230 CT scans were also included as non-infected CT images, so from the second dataset, 2482 CT scans were obtained. This dataset was collected from multiple hospitals in Sao Paulo, Brazil. Train and test sets were obtained from a random permutation of two datasets aggregated. Depending on the task and number of data centers for federated learning, the data were permuted in multiple centers in a random manner. You can see sample COVID infected and Non-Covid images... The study formulation is binary classification, i.e., we develop the model the classify COVID-19 infected images with normal healthy CT scans. The distribution of data is shown in the table below

Data	Class	Dataset	Train data	Test data
COVID-19	Kaggle	1252	1451	150
	Tongji	349		
Non COVID-19	Kaggle	1230	1477	150
	Tongji	397		

Table 1: A test caption

4.2 Preprocessing

The images are first resized into 256×256. Then as a preprocessing, the images were cropped in random sizes and in various areas. The cropped area is a random subsection of the original image. The lower and upper bounds for the subsection were 0.5 and 1, meaning that the selected area to crop could vary from half of the image and more. Then the images were resized into size 224×224. The interpolation method for resizing is bilinear. Images were randomly flipped as a data augmentation method. Then the pixel intensities are normalized, with mean = 0 and std of 1.

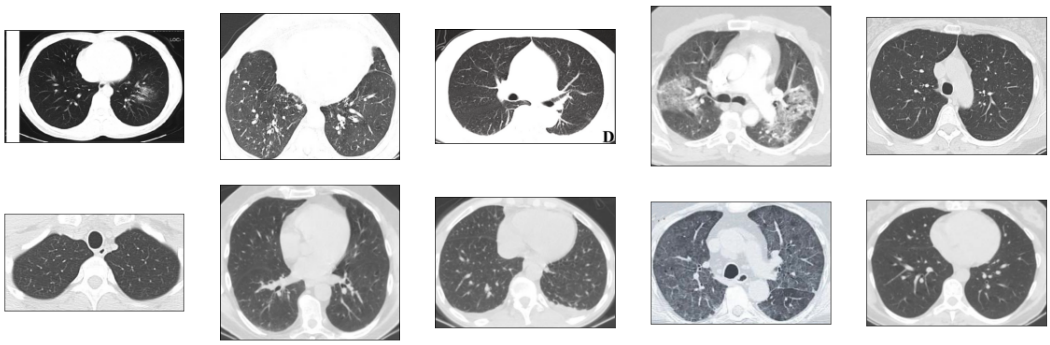

Figure 1: Sample COVID-19 infected and Non-Covid CT scans.

4.3 Training

The model used to classify COVID-19 from healthy patients is ResNet-18. ResNet-18 consists of 16 convolution layers, two downsampling layers, and fully connected layers(FC). The input image size of ResNet is 224×224. In addition to the first convolution layer, the convolution kernel size is 7×7, and the other layers are 3×3. After average pooling the feature map of the last convolution layer, an eigenvector is obtained by full connection, then the classification probability is obtained by normalization with Softmax. The convolution layer that outputs the same size feature map has the same number of filters. Two convolution layers of the same color form a residual block. Shortcut connections are those skipping two layers ..ResNet-18 will obtain an eigenvector that contains multiple probabilities, which are used to indicate that the input image belongs to a certain class, and the class with the highest probability will be the output finally. The number of input channels of the fully connected layer must be fixed, so the input image of ResNet-18 needs to be a fixed size. The data was split randomly into separate data stations, and federated models assigned to each data center had access to only data from that center. There were submodels and their datasets.

There were various settings in which the data was split in a various number of data centers. Here is a sample image of a setting with eight hospitals, showing a number of healthy and covid samples in each.

5 Results

5.1 Evaluation criteria

In order to evaluate and compare federated learning models, specificity and sensitivity are the abilities of a model that how correctly the model identifies a subject with disease and without a disease. In our case, it is critical to detect a COVID-19 patient as missing a COVID-19 patient can have disastrous consequences. Evaluation metrics The formulas of the measures are given as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

$$Precision = \frac{TP}{TP + FN} \tag{16}$$

Where 'TP' (true positives) are images that were correctly classified by the network, 'TN' (true negative) is negative COVID-19 images that were correctly labeled. 'FP' (False positive) are images that had COVID-19 but were labeled as negative. 'FN' (False negative) are COVID-19 images that were not detected by the classifier and reported as non-COVID A medical diagnosis-based system needs to have a high recall.

5.2 Experiment results

Client experiments

A Singel institution or centrally gathered dataset was examined as the baseline method for comparing federated learning algorithms. The training set had high accuracy. Centralized data sharing was performed and had 81 percent accuracy. Training for more rounds didn't have an impact on final test accuracy since the model was already learned the data. There were various criteria to compare different federated learning algorithms. The federated learning algorithms were examined under different conditions. The average score in each of their conditions could be a way to compare them—the average scores in F1, FedSGD, CWT, and SWT t. The results for CDS are also included as a benchmark ground truth. Methods like SWT, CWT, and FedSGD show higher performance than CDS and FedAvg.

Effect of the number of clients FL models were examined under different settings and with various hyperparameters. To investigate the number of clients on the networks' performance, the results for each FL setting were examined under different hyperparameters, and the average results were reported. As an observation, test accuracy drops significantly as the number of clients increases, this relation comes from the fact that with the total amount of data being the same, each client has fewer data and will train the less accurate model. This strong relationship was not observed in Precision and recall compared to the accuracy of the models.

Number of clients	FL model	Recall	precision	Accuracy
2	FedAvg	0.890	0.969	0.782
	FedSGD	0.984	**1.00**	**0.994**
	CWT	0.976	**1.00**	0.988
	SWT	0.994	0.991	0.992
	MNFL	**0.964**	0.972	0.982
4	FedAvg	0.874	0.969	0.728
	FedSGD	0.929	0.973	0.931
	CWT	0.785	0.789	0.981
	SWT	0.955	0.947	0.952
	MNFL	**0.924**	**0.842**	**0.982**
10	FedAvg	0.627	0.705	0.600
	FedSGD	**1.00**	**1.00**	0.922
	CWT	0.938	0.991	**1.00**
	SWT	1.00	1.00	0.888
	MNFL	**1.00**	**1.00**	**1.00**
20	FedAvg	0.701	0.966	0.532
	FedSGD	0.947	0.967	0.884
	CWT	0.934	1.00	0.842
	SWT	0.795	0.862	0.823
	MNFL	**0.975**	**0.962**	**0.923**

Table 2: A test caption

Effect of number of rounds One round is generally defined as one time in which the model passes all the clients. In FL models which the model can pass each client multiple times, this hyper-parameter can be defined, and as our results suggest, this hyper-parameter can have a significant effect on the performance of the FL model.
Generally, as the rounds increase, the performance gets better, but due to distributed structure of the network, the accuracy and loss function have volatile behavior.

Effect of number of clients As discussed in the Methods section and as the literature suggests, [21], FedSGD is a generalization of the FedAvg method, in which the number of participants each round is a subset of total participants. Choosing a subset of participants generally improves training time and also the overall model performance. In nearly all of the experiments, choosing a subset of clients led to improved performance compared to when all the clients are included.

Bias Having less number of clients, SWT and CWT models perform nearly similar, but as the number of clients increases, CWT surpasses the SWT performance. This observation is due to an effect called *catastrophic forgetting* [22]. In FL models in which the clients are trained sequentially, global models are more likely to have a bias in favor of the latest client. This could be seen in training procedures for domain adaptation/transfer learning, where an initial model is re-trained on a new dataset. This model performs quite well on the latest dataset it was since it is fine-tuned on that specific data distribution. Since the test set is a general domain data, catastrophic forgetting leads to a decrease in model performance. One thing to note is that as the number of clients increases, bias is more likely to happen.

The reason is that in this experiment, clients are made by random sampling of one large dataset. As the number of clients increases, each client is less likely to display the distribution of the main dataset. The network has a bias in the latest client. It deviates from the main dataset more. Another thing is that although CWT and SWT are both sequential and bias exists, but since in CWT, a model is passed through each client more than once, the bias can be diminished. Figurefig:CWT bias shows the overall accuracy of the CWT model in two different experiments. As the model passes clients, in the first two rounds, the model is not performing well, but in the next cycles, the accuracy reaches its plateau.

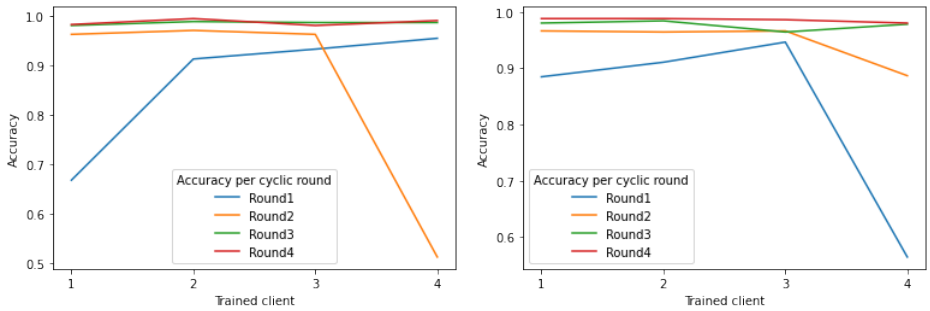

Figure 2: Accuracy of CWT algorithm as the model passes each client

Network being trained on a client with a distribution much different than the distribution of other clients, will results in a drop in performance. In methods like FedAvg, since the results are averaged, this effect can be negligible . However, in sequential methods like CWT, if there is an outlier client, it matters that when the client will reach that client. For example in figure fig:CWT bias right, the fourth client has a different data distribution and the accuracy is dropped when reaching that client. Either more rounds are needed or the order of clients should be changed so that the outlier client is not the one which mostly affects the global model.

Network load How much communication an FL network needs in order to perform. The amount of communication is calculated by how many times the model is sent/received by the involving parties multiplied by the size of the deep learning model. The table below shows the average computation cost for each FL model in various settings. The computation costs are calculated by the total duration of training time. Higher internal training epochs and the number of rounds tend to have a higher computation load. Computation loads are higher for CWT in comparison to other FL networks under the same network settings.

Number of clients	FL model	Avg. computation cost
2	FedAvg	0.890
	FedSGD	0.984
	CWT	0.976
	SWT	**0.994**
4	FedAvg	0.874
	FedSGD	0.929
	CWT	**0.985**
	SWT	0.955
10	FedAvg	0.627
	FedSGD	**1.00**
	CWT	0.938
	SWT	0.924
20	FedAvg	0.701
	FedSGD	**0.947**
	CWT	0.934
	SWT	0.795

Table 3: Avg. computation cost for each FL model

6 Conclusion

One problem in comparing these types of algorithms is how to change their hyperparameters so that they are actually comparable. This paper introduced MNFL, a peer-to-peer federation learning environment for distributed training. In contrast to the traditional FL with a server, our framework does not rely on the central server itself to coordinate the training process. We presented a proof-of-concept study that handled the demanding tasks of whole-brain segmentation and trained a complex fully convolutional neural network in a decentralized way. Experiments with showed that MNFL outperformed other FL methods with different experimental settings.In scenarios where the number of customer training scans is not equal, the margin can be up to 7% cube points. Overall, MNFL not only solves central server-dependent issues, but also enables more robust training of clients through highly dynamic updates, comparable to models trained with data pooled through clients. FL has developed as a new paradigm for cooperatively training models among several customers while maintaining anonymity. Personalization in FL is critical for the future trend because to the diversity of users (e.g., statistical and systematic variability, etc.). Our technique MNFL not only overcomes the obstacles of homogenous model constraint, which may considerably minimize communication cost during training, but it also increases training efficiency through a parameterized update mechanism with no additional computing overhead at the client side. This study has the potential to enable multiple devices to train ML tasks jointly using customized neural network architecture.

References

[1] A. Waheed, M. Goyal, D. Gupta, A. Khanna, F. Al-Turjman, and P. R. Pinheiro, "Covidgan: data augmentation using auxiliary classifier gan for improved covid-19 detection," *Ieee Access*, vol. 8, pp. 91 916–91 923, 2020.

[2] G. D. Rubin, C. J. Ryerson, L. B. Haramati, N. Sverzellati, J. P. Kanne, S. Raoof, N. W. Schluger, A. Volpi, J.-J. Yim, I. B. Martin, *et al.*, "The role of chest imaging in patient management during the covid-19 pandemic: a multinational consensus statement from the fleischner society," *Radiology*, vol. 296, no. 1, pp. 172–180, 2020.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[6] S. Jinnai, N. Yamazaki, Y. Hirano, Y. Sugawara, Y. Ohe, and R. Hamamoto, "The development of a skin cancer classification system for pigmented skin lesions using deep learning," *Biomolecules*, vol. 10, no. 8, p. 1123, 2020.

[7] K. Muhammad, S. Khan, J. Del Ser, and V. H. C. de Albuquerque, "Deep learning for multigrade brain tumor classification in smart healthcare systems: A prospective survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 507–522, 2020.

[8] Q. Jin, Z. Meng, T. D. Pham, Q. Chen, L. Wei, and R. Su, "Dunet: A deformable network for retinal vessel segmentation," *Knowledge-Based Systems*, vol. 178, pp. 149–162, 2019.

[9] Y. Gordienko, P. Gang, J. Hui, W. Zeng, Y. Kochura, O. Alienin, O. Rokovyi, and S. Stirenko, "Deep learning with lung segmentation and bone shadow exclusion techniques for chest x-ray analysis of lung cancer," in *International Conference on Computer Science, Engineering and Education Applications*. Springer, 2018, pp. 638–647.

[10] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep learning for identifying metastatic breast cancer," *arXiv preprint arXiv:1606.05718*, 2016.

[11] M. A. Azad, J. Arshad, S. M. A. Akmal, F. Riaz, S. Abdullah, M. Imran, and F. Ahmad, "A first look at privacy analysis of covid-19 contact tracing mobile applications," *IEEE Internet of Things Journal*, 2020.

[12] H. Xu, L. Zhang, O. Onireti, Y. Fang, W. J. Buchanan, and M. A. Imran, "Beeptrace: Blockchain-enabled privacy-preserving contact tracing for covid-19 pandemic and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3915–3929, 2020.

[13] T.-C. Chiu, Y.-Y. Shih, A.-C. Pang, C.-S. Wang, W. Weng, and C.-T. Chou, "Semisupervised distributed learning with non-iid data for aiot service platform," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9266–9277, 2020.

[14] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.

[15] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[16] Z. Zhou, S. Yang, L. Pu, and S. Yu, "Cefl: online admission control, data scheduling, and accuracy tuning for cost-efficient federated learning across edge nodes," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9341–9356, 2020.

[17] W. Zhao, W. Jiang, and X. Qiu, "Deep learning for covid-19 detection based on ct images," *Scientific Reports*, vol. 11, no. 1, pp. 1–12, 2021.

[18] M. J. Horry, S. Chakraborty, M. Paul, A. Ulhaq, B. Pradhan, M. Saha, and N. Shukla, "Covid-19 detection through transfer learning using multimodal imaging data," *IEEE Access*, vol. 8, pp. 149 808–149 824, 2020.

[19] Y. Pathak, P. K. Shukla, and K. Arya, "Deep bidirectional classification model for covid-19 disease infected patients," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[20] L. Wang, Z. Q. Lin, and A. Wong, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images," *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.

[21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[22] N. Shoham, T. Avidor, A. Keren, N. Israel, D. Benditkis, L. Mor-Yosef, and I. Zeitak, "Overcoming forgetting in federated learning on non-iid data," *arXiv preprint arXiv:1910.07796*, 2019.