DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Federated Learning for Medical Applications

Soham Mazumder

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Federated Learning for Medical Applications

Föderales Lernen für medizinische Anwendungen

Author:	Soham Mazumder
Supervisor:	Prof. Dr. Nassir Navab
1st Advisor:	Magdalini Paschali, M.Sc.
2nd Advisor:	Matthias Keicher, Dipl.-Ing.
Submission Date:	January 15th, 2020

I hereby declare that this master's thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, January 15th, 2020 Soham Mazumder

Acknowledgements

First and foremost, I would like to thank my advisors, Magda Paschali and Matthias Keicher for their continuous assistance. They helped to formulate the objective of my thesis, contributed with important ideas and lent me support at my low points. This work would not have been possible without them. I would also like to thank Prof. Nassir Navab for giving me this opportunity to pursue research at his chair. Special thanks go to Abhijit Guha Roy, who provided me with data and innovative ideas.

"Without data you're just another person with an opinion."

-W. Edwards Deming

Abstract

The concept of federated learning has revolutionised decentralised machine learning. It enables the learning of a single global model from data distributed among different sources, without the need for aggregation and while preserving the privacy of each data source. Federated learning is being used in modern mobile phones and smart devices to learn various tasks. The idea of privacy-preserving learning is extremely suitable for the field of medicine. Since medical records are inherently confidential, federated learning can be applied to its full potential.

Based on a review of the literature, it is evident that there has not been much advancement towards the use of federated learning in medical tasks. In this study, we analyse the viability of federated learning to solve tasks like classification and segmentation, using datasets procured from hospitals and medical institutions. Our quantitative results for classification illustrate that the best performance of federated classification (Accuracy = 82.5%) is very close to the baseline of a model trained with centralised data (Accuracy = 83%). We can achieve similar results for the federated segmentation tasks. Our best performance (Dice = 0.847) is similar to our centralised model baseline (Dice = 0.867). We also compare federated segmentation with two alternative decentralised learning methods and find they are not able to outperform federated learning. The last experiment we conduct is federated segmentation with multiple datasets to evaluate the performance of our model in a real-world setting.

Contents

Part I.

Introduction and Related Works

1. Introduction

In medicine, there are various manual processes like handwritten patient records, diagnoses, and writing lab charts. Such mundane tasks can be easily automated, leaving the doctors to concentrate on more important aspects, like surgery and medical research. Electronic Medical Record is once such step towards automation, where all patient data is digitised and archived properly. In order to use machines to help the clinician make better decisions regarding diagnoses and treatment, and make use of the digitised records, we need Machine Learning. Machine Learning algorithms are commonly used as classifiers. The process involves feeding handcrafted features extracted from an image to a machine learning model like Multi-Layer Perceptron [43] and Support Vector Machines [52]. From these features, the model learns decision boundaries to classify the objects in the image. Computer-aided detectors (CAD) [45, 10] became popular in the 2000s. Medical images like MR scans and X-rays were digitised and fed into the system. The CAD used machine learning to detect and segment specific objects in an image automatically.

However, there remains the step of feature extraction where the involvement of human experts is required. So naturally, the next step in advancement involves removing any human element from the process by teaching the model to extract its features from the image. This concept is now widely known as Deep Learning. The deep learning approach consists of a neural network, which acts as a model classifier that can take images as input and provide classification results. The most efficient image classifier is the convolutional neural network [24]. Such networks use convolutional filters to transform the input and extract features from it. The turning point of deep learning from where it started to garner huge popularity is with AlexNet [21], which outperformed all existing algorithms by a huge margin to win the 2012 Imagenet Challenge. Many more state-of-the-art architectures followed like VGG [49], ResNet [15] and InceptionNet [51].

Deep Learning enabled the automation of many medical tasks. We now have systems that use deep learning algorithms to detect cancerous growth from a mammogram [1, 47], classify skin lesions [2], detect diabetic retinopathy from retinal images [6] to name a few.

Deep learning is a data-driven process. With more data, such artificially intelligent systems can provide better information to patients, and help the clinicians more to make the best decision. Deep learning reduces the probability of false-negative results and also increase the speed of getting the results. However, most current, state of the art deep learning models need the entire dataset to be stored centrally, all in one place. Medical data, on the other hand, is different from other data. Medical data is primarily obtained from hospitals. As a result, they are private and confidential. So, it is required to preserve a certain level of privacy when using medical data. Furthermore, since these data cannot be released outside of hospitals, it is not feasible to collect them in a centralised storage location. As a result, aggregation and mixing of data from different sources are restricted, which puts a limitation on the use of machine learning models on medical data.

We can overcome this serious problem by a method called Federated Learning. In federated learning, any deep learning algorithm can be trained using distributed data sources to learn a single model while maintaining the privacy of the data from different sources. Therefore, we can use data from many different hospitals and medical institutions without aggregating the data in central storage.

In this thesis, we analyse the feasibility of federated learning in the medical domain by applying it in two different challenging tasks, namely skin lesion classification and whole-brain segmentation. The results from our experiments validate the advantage federated learning has over traditional deep learning approach, especially in the medical domain.

1.1. Federated Learning

We are now in the age of electronics, with almost fifty percent of the world population having access to a smartphone. In addition to phones, there are wearable devices and autonomous vehicles. All these devices are an abundant source of data, generating huge amounts every day. Now, with the ever-growing computational power of these devices, it is a realistic idea to store and learn the models locally. This concept is known as Edge Computing. Edge computing has been explored for a few decades. Much research has been done in the direction of query processing in sensory networks [9] and fog computing [4]. In the field of user modelling, there have been recent approaches that try to train machine learning models in a central server and then serve them to the local devices [22].

Recently, with an increase in power and storage of edge devices, it has become feasible to use local resources to training device specific models. The next step would be to acquire a global model which is an aggregate of all local models. This idea led to the inception of Federated Learning (FL).

FL is first proposed in [33] where it was utilized towards using the massive amount of textual data stored in the smartphones of Android users around the world to build natural language processing models. The primary constraint to this approach was the private nature of the data. Since these local datasets in each smartphone are confidential user data, they could not be aggregated together. As a solution, they came up with the FL pipeline as seen in Fig 1.1.

The authors considered each user's smartphone as an individual client, with every client coordinated by a central server. The server maintained a neural network model, which is sent to each client as individual copies. Each client then trained their copy of the central model on their local dataset. The trained, parameters of the client models were then sent to the server. At the server, these parameters from each client were aggregated together to get the updated central model. This process was repeated multiple times to achieve the best possible accuracy. This learning process enables the system to learn a global model

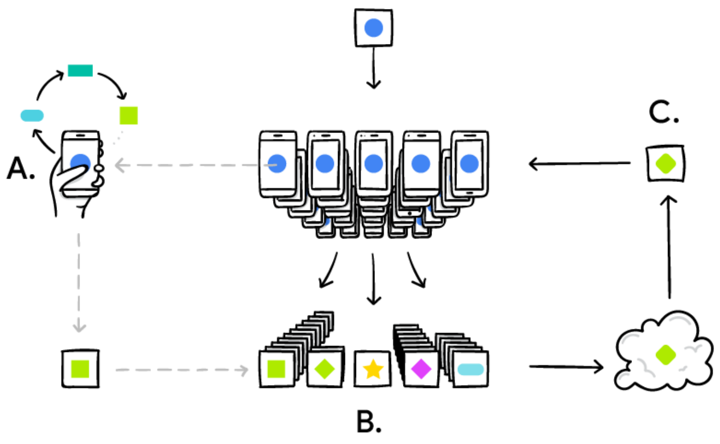


Figure 1.1.: Each user's smartphone trains a model on it local data (A). The model updates from many users are aggregated (B), to get the averaged update (C) which is applied to a global model. Then the entire cycle is repeated. [32]

from a distributed dataset while preserving the privacy of each sub-dataset. The data privacy is preserved since the server model has access only to the parameter updates of the client models and not directly to the data. McMahan et al. [33] also introduced Federated Stochastic Gradient Descent and Federated Averaging algorithm, two types of optimization schemes for the federated setting.

An example application of the paper would be the task of sentence completion in smartphones (cf. 1.2). In order to preserve the privacy of the textual data present in smartphones, a language prediction model is learnt in a distributed manner. The model updates learnt by each smartphone are then sent to the global server, where they are aggregated together to form a global model.

This paper was a breakthrough in the field of decentralized machine learning. However, it had its share of flaws. This paper considered the distributed datasets to be simple, clean and belonging to the same distribution. It ignored the possibility of heterogeneous datasets, variable dataset size and the real-time modification to client datasets. The authors also avoided many possible practical challenges of system design like client unavailability or non-responsive clients and corrupted updates to the server.

1.2. Major Challenges

In this section, we list the main roadblocks of the FL framework. These challenges distinguish federated learning from other distributed learning methods which are commonly

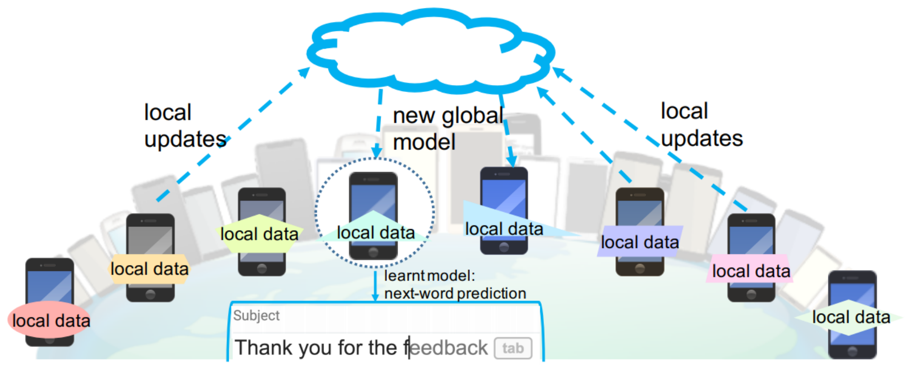


Figure 1.2.: Training a sentence completion model using FL [27]

deployed in data centers.

1.2.1. Communication Cost

Communication can be a major bottleneck in federated frameworks. Since in FL we prioritise privacy, it is not possible to transmit raw data among the devices. Moreover, a federate setting can comprise hundreds to thousands of devices, and so it is naturally preferable to run local computations. It then becomes imperative to optimize communication schemes. An example would be to send model updates. Two key points to consider to improve communication methods are:

1. Reducing number of communication rounds between client and server.
2. Reducing size or amount of data sent in a communication round.

1.2.2. Systems Heterogeneity

The performance of each device in a federated setting can be varied. Realistically, different devices will have variable computational power, storage limit and connectivity capabilities. Furthermore, not all devices are guaranteed to be available all the time. An active device can be unreliable and drop out suddenly due to technical constraints. These issues lead to challenges such as,

1) Straggler Mitigation

Devices which take more time to complete their computation, maybe due to less power or inferior hardware, are called stragglers. Stragglers force other devices to wait for it to complete its computation, thereby wasting communication time of the system. Mitigation strategies are required to handle stragglers.

2) Fault Tolerance

In a federated network, a device can suddenly stop its computation due to internal error. The network ought to be designed such that it continues operating properly, irrespective of a faulty device.

As such, federated frameworks must be built to be robust to these challenges.

1.2.3. Statistical Heterogeneity

A common but serious issue in federated networks is the variation in the distribution of data in each device. The data in each device is not independent and identically distributed (Non-I.I.D.). This nature of the distributed data violates the independent and identically distributed (I.I.D.) assumptions in most optimisation schemes that are commonly used. Non-I.I.D. results in poor convergence of the training method leading to sub-par performance of the network. A standard method to tackle this problem is to use device-specific modelling.
There are multi-task learning frameworks, which learn multiple models, specific to each device. Such methods are different from federated learning where we learn a single global model.
There are also meta-learning approaches [26] which enable device-specific modelling.

1.2.4. Privacy

Ultimately, the main idea of FL is to preserve the privacy of the device data. By only sharing model updates, federated frameworks assure device-level data privacy. However, security can be compromised via the transmitted model updates. Recent methods try to enhance the privacy of federated networks by using various methods like secure multi-party communication (SMC) or differential privacy [11].
SMC is an encryption scheme which allows for encrypted averaging of device updates. This method prevents the reconstruction of device data from the updates.
Differential privacy enables the sharing of model update without revealing the device from which the update originated. Such methods improve the privacy of data in a federated network, but the added security comes at the cost of inferior performance of the model.

1.3. Federated Learning with Medical Data

In the framework of FL, one can replace multiple devices with hospitals and medical institutions. Medical data by nature is extremely sensitive, and hospitals are bound by legal and ethical constraints to keep their data locally. It is therefore vital to ensure the security and confidentiality of medical data while using it for a machine learning task. Thus, the concept of federated learning can be effective.

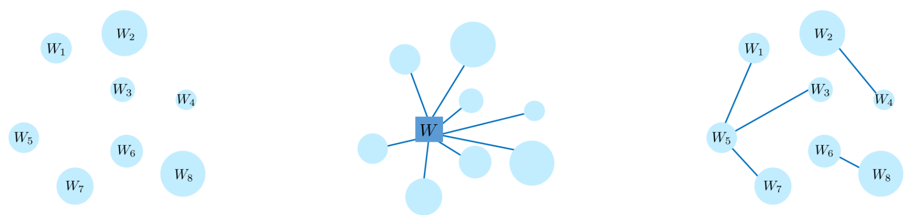
The use of Federated Learning in the medical domain is relatively new. Some of the earliest works were on textual data, namely on patient record datasets. Recent works [17], [18], [30] implemented federated learning on electronic medical records of patients for classification and regression tasks. A more advanced application was implemented by [46] and [41] where the federated framework was applied to the task of segmentation of brain MRI scans.

2. Related Works

The challenges in federated learning, as stated before, range from traditional problems like privacy concern, distributed machine learning, and optimisation to harder problems like statistical and systems heterogeneity which are peculiar to federated learning. In this section, we explore the recent works that have tried to address these challenges.

2.1. Statistical Heterogeneity

In the federated setting, there are mainly three modelling approaches (cf. 2.1). The choice of modelling depends on the type of data, network architecture and the type of task.



(a) Learn personalized models for each device; do not learn from peers.

(b) Learn a global model; learn from peers.

(c) Learn personalized models for each device; learn from peers.

Figure 2.1.: Three data modelling approaches - (a) device-specific model without sharing of data, (b)FL approach, (c)a peer-to-peer approach. [27]

A significant drawback for FL is the non-Independent and Identically Distributed (Non-I.I.D.) nature of the client datasets. In [55], the authors focused on the challenge of using Non-I.I.D. data for FL. Initially, it was highlighted that highly skewed Non-I.I.D. data significantly reduces the accuracy in an FL system. This drop in performance occurs due to the weight divergence of different client training. Their proposed method overcomes this issue by initially training the global model on a small subset of data, globally shared between all the clients' devices, that is also representative of the distribution of the entire population. So, with a better initialisation, the client training would converge even with Non-I.I.D. client data. The paper successfully showed how to handle Non-I.I.D. data in Federated systems using the concept of transfer learning. The downside of this method is that some data has to be centrally located, which defeats the truly distributed purpose of the federated system.

In Smith et al. [50], the authors introduced an optimisation framework specifically for the federated setting. The framework facilitated the learning of separate but related models for each device while leveraging a shared representation through multi-task learning. This allowed for personalisation of each device. However, this method is not scalable to larger networks and can only optimise convex objective functions.

Another factor to consider while modelling federated data is fairness. By solving an aggregate loss naively, we risk inculcating a bias in training towards devices which may have more data, or more commonly occurring data. As such, some works have proposed weighting the individual device to reduce variance in their performance. [18] simply trained each device for a varied number of epochs based on their loss. In Li et al. [28], authors proposed an augmented objective termed q-Fair Federated Learning (q-FFL), in which a device with larger loss is given a larger weight, to encourage less variance among the devices. [35] tried to tackle device variance by optimising the distribution of the centralised model represented as a mixture of the client distributions. Their results show that fairness among the devices is also attained.

The Non-I.I.D. nature of the client data in federated learning can adversely affect the convergence of the global model. So naively using Federated Averaging, which was introduced in [33] is not enough. Sahu et al. [44] introduced a small modification to FedAvg so that the global model converges, even in case of severe Non-I.I.D. datasets. In this method, the number of iterations per device is decided based on its system capabilities. Further, the loss is augmented by a term that ensures that the individual device weights do not diverge very much from the global weights.

Since the data of each device can be varied, the models trained in federated setting may not converge due to domain shift problems. Peng et al. [38] introduced domain adaptation into the federated framework to improve convergence. They implement three steps to ensure domain adaptation- a dynamic attention mask, adversarial alignment of each device and representation disentanglement. Yurochkin et al. [54] proposed a Bayesian non-parametric framework for federated learning to aggregate those weights which have a similar feature extraction signatures.

2.2. Privacy

McMahan et al. [33] showed a fundamental way to train models on distributed datasets effectively without going into details of privacy. Geyer et al. [14] focused more on client privacy. They wanted to ensure that the learning process does not reveal the participation of particular clients during decentralised training. So they aimed to protect a client's entire data set against any form of differential attack. They proposed a randomised mechanism comprising of two parts - Random Sub-Sampling of the clients and Distortion of Weight Updates. For each communication round, a subset of clients was chosen randomly from the entire pool of clients. Then during update transmission, the model updates were distorted so that the clients which participated in that communication round remain anonymous. This method would help to preserve differential privacy in federated learning. The authors showed through experiments that they needed a large number of participating clients to match the performance of [33] while preserving client- level differential privacy. However, this condition of many clients is not always fulfilled and in case of a smaller number of clients this method performs quite poorly. [29] looked into the feasibility of applying differential privacy to a federated medical application, without degrading the performance of the network.

Another way to preserve privacy while learning from distributed data is to use secure multiparty computation (SMC). In SMC, a value is split into multiple shares, and each share is sent to different parties or devices. These shares act as private keys, so all the devices must agree to perform decryption. However, such methods are often unscalable and degrade the performance of the network. Some instances where SMC was used in machine learning can be found here [5].

To address privacy concerns of distributed methods, Papernot et al. [36] proposed the Private Aggregation of Teacher Ensembles or PATE. In this method, the learned representations from an ensemble of teacher networks are transferred to a student model. The teachers are analogous to the various devices, and their trained updates are aggregated into the student (or global) model. This paper also augments the aggregation process with noise to make the student more robust.

2.3. Medical Applications

The use of Federated Learning in the medical domain is relatively new. Even though it is very attractive to apply federated learning to medical tasks, medical data has its share of challenges. Scarcity of medical data is a major problem, which leads to poorly trained models. Medical data also have different modalities which make any task more challenging. Furthermore, the need for privacy and confidentiality of medical data is paramount.

Huang et al. [18] applied FL on a patient medical record dataset, MIMIC-III [19]. The authors of this paper tried to tackle the problem of heterogeneity of client datasets. They introduced a method called Loss-based Adaptive boosting, which compared the cross-entropy loss of each client in the current iteration with the median loss of all clients in the previous iteration to decide how many local epochs a client should be trained. However, epoch loss of a client is not the best choice for its stopping criteria, since loss can fluctuate even when a model is still learning.

Another work [17] introduced a community based approach to FL (cf. 2.2). The authors tried to find meaningful clusters within the distributed data to group them into different communities, such as disease type and hospital location. Privacy-preserving representations of patient data were clustered using K-means. Afterwards, each hospital individually trained five community models. The number of communities depended on the number of clusters. Then for each community, a model was learnt in a federated manner, and the updated parameters were sent to the server for aggregation by weighted averaging.

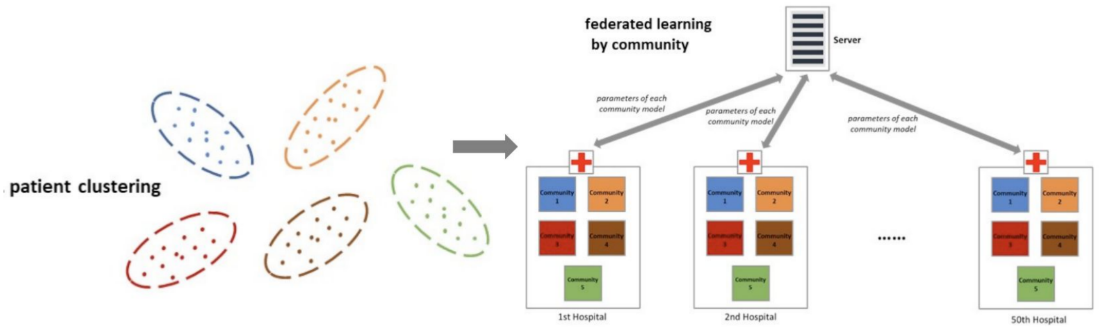

federated learning by community
Server
patient clustering
parameters of each community model
parameters of each community model
parameters of each community model
Community 1
Community 2
Community 3
Community 4
Community 5
......
1st Hospital
2nd Hospital
50th Hospital

Figure 2.2.: Community Based FL - patient data is grouped into clusters based on features, and then each hospitals learn a model specific to its data. [17]

They presented their results on the eICU collaborative research database [39] for Mortality prediction and Hospital Stay Time prediction. The main problem in this paper is in the clustering phase. If the data in different hospitals are very diverse, finding meaningful communities to group the data can be challenging. Also, choosing the number of communities is a major issue.

Dianbo et al. [30] modified conventional FL to tackle heterogeneous distributed data. Their idea was to train a global model using all data sources in a normal federated manner. Then they froze the first few layers of the model and further trained the rest of the model using data from specific data sources to specialise it for each client dataset. They also showcased their results on the eICU collaborative research database [39]. This method showed promising results, but it diverged from the FL framework. We do not get one global model, but multiple different models optimised to each client. This inherently violates the main principle of FL for having a single global model for a distributed dataset.

Sheller et al. [46] were the first to apply FL on a multi-institutional medical image dataset. They compared their implementation with other distributed learning algorithms, namely Institutional Incremental Learning (IIL) and Cyclic IIL. In IIL, institutions train a shared model in succession. Each institution trains the model only once and may train the model however it chooses. A considerable drawback of IIL is catastrophic forgetting. Cyclic IIL is implementing IIL for multiple cycles such that each institution can train the shared model multiple times. Cyclic IIL doesn't suffer from catastrophic forgetting but needs massive overhead for validation. The authors experimented on the BraTS Dataset [34]. They successfully showed the superiority of FL over the other methods. The authors of this paper were pioneers of using FL on a medical image dataset. However, their implementation of the federated framework was largely inspired by [33]. So, this method also suffers from the problems present in [33].

One of the most recent paper to apply federated learning in the medical domain is Brain Torrent [41].

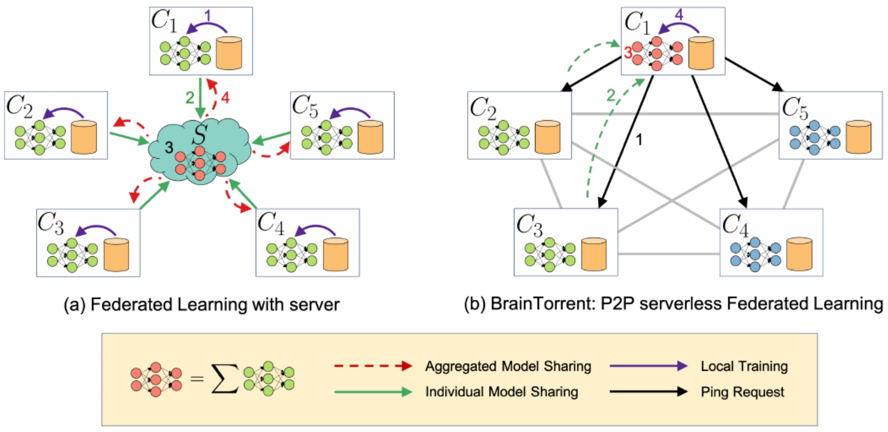

(a) Federated Learning with server
(b) BrainTorrent: P2P serverless Federated Learning
Aggregated Model Sharing
Individual Model Sharing
Local Training
Ping Request

Figure 2.3.: Compares FL (a) with their peer-to-peer FL (b) [41]. The FL has a server which coordinates the clients. Brain torrent on the other hand work in a peer-to-peer environment, where all clients communicate with each other.

It proposed a highly dynamic peer-to-peer environment, where all clients directly interact with each other. A client sends a ping request to all other clients for checking their versions. Clients that have newer versions send their updates after the request. An aggregated model (red in 2.3) is formed by combining the updates from the clients and afterwards is fine-tuned on local data of the specific client. There is no dependency on the central body. Their experiments were focused on the task of whole-brain segmentation. The authors aimed to remove the vulnerability of the server. Failure of the server would result in a crash of the entire system. The author presented their results on the Multi-Atlas Labelling Challenge (MALC) dataset [23]. The disadvantage of this peer-to-peer framework is high communication cost. Since each client needs to interact will all the other clients to communicate the updates, the entire process becomes very costly. This would also cause the training to be much slower compared to the framework in [33].

The privacy aspect of federated learning has also been addressed within the medical community. A recent paper [29] explored the idea of applying differential privacy to the federated framework to ensure data protection (cf. 2.4). The results from the paper show that there exists a trade-off between network performance and privacy costs.

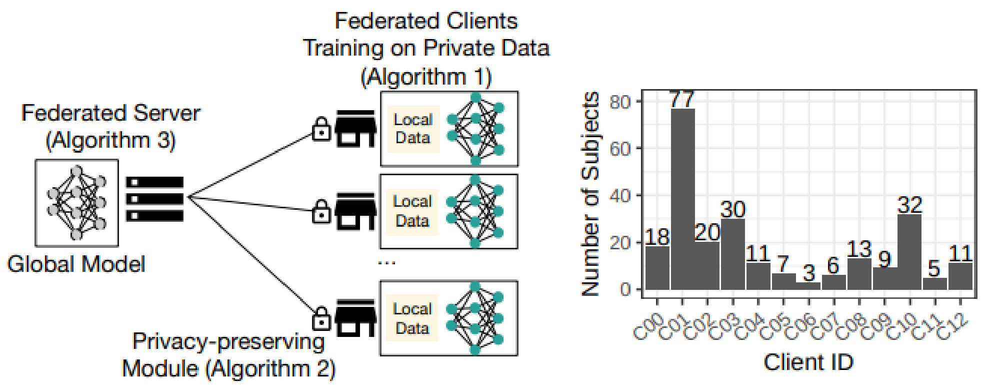


Figure 2.4.: FL with anonymous client participation, which preserves the privacy of each client data. [29]

Part II.

Methodology

3. Federated Learning

In a classification problem, we train a machine learning model to identify the category of a new observation using a training set of data comprising of instances (x_i) whose category (y_i) is known. The model contains parameters (weights and biases) which are learnt through the course of the training. These parameters are often denoted by w. During training, the model computes a loss $l()$ of the prediction on a data-point (x_i, y_i), obtained from a model with parameters, w as :

$$f_i(w) = l(x_i, y_i, w) \tag{3.1}$$

For FL, the data is partitioned over K clients, $(K_1, K_2, ...K_3)$ where K_k refers to the k^{th} client. P_k denotes the set of indices of data points of client k and so amount of data in k^{th} client is, $n_k = |P_k|$.
So, for a client k the loss is denoted by,

$$F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} f_i(w) \tag{3.2}$$

After each client has completed their computation, we then aggregate the loss of all clients K, using weighted averaging. The weights correspond to the size of the dataset present in each client. Then the at the server side, the problem objective is given by,

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{3.3}$$

3.0.1. Federated Algorithm

The federated learning process is explained in Figure 3.1. Each communication round consists of steps 1 to 4. These steps are repeated until a good performance is achieved.

1. K clients are selected. The server sends the current global model to each client.
2. Each client trains its copy of the model using the local dataset. The updated model parameters are obtained after finishing computation.
3. These updated parameters are sent by each client to the server.

4. At the server, the updates are aggregated using weighted averaging,
$W_{server} = \sum_{k=1}^{K} \frac{n_k}{n} w_k$.
The W_{server} is then applied to the global model.

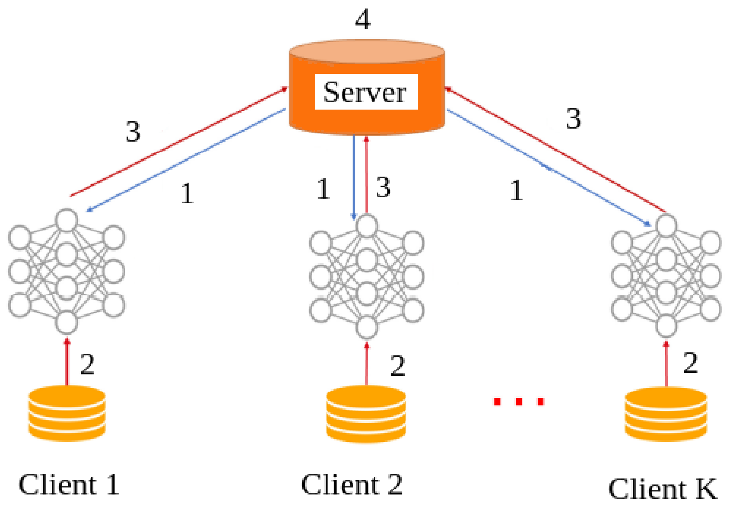


Figure 3.1.: Overview of the FL framework. Each communication round consist of 4 steps, 1) A copy of global model sent to all clients, 2) Clients perform local computation to update local model parameters, 3) All clients upload parameter update to server, 4) Updates are aggregated and applied to global model.

3.0.2. Federated Optimization Schemes

McMahan et al. [33] introduced two optimization methods for FL:

- Federated Stochastic Gradient Descent (FedSGD): In this method, for each communication round, each client computes for one local epoch and without taking batches of the data but instead taking entire local dataset together.
- Federated Averaging (FedAvg): In this method, local computation is increased by increasing local epochs and decreasing batch size. The FedAvg algorithm is defined below.

Federated Averaging Optimization

Algorithm 1: Federated Averaging. There are K clients, local batch denoted by B, local epoch is E and η is learning rate.

```
Server
for each communication round t do
    for each client k do
        $w^k_{t+1}$ = ClientUpdate$(k, w_t)$
        $w_t + 1 = \sum_{k=1}^{K} \frac{n_k}{n} w^k_{t+1}$
    end
end
ClientUpdate$(k, w)$
for i in local epoch 1, 2, ..., E do
    for batch $B$ in dataset of size $n_k$ do
        $w = w - \eta \nabla l(w; b)$
    end
end
return w
```

In algorithm 1 for a client, we train the dataset for multiple epochs. Also, we take the data in batches, so the model is trained for multiple iterations in one local epoch. After the clients are trained, they send their updated parameters, w to the server. At the server, these updates are aggregated and applied to the global model.

4. Federated Classification

One of the most common tasks in machine learning is Classification. It is a supervised method of training a machine to identify the category of an object. Classification of medical images is a critical area of research that garners expanding attention from both the research sector and the medical community. So our first application of the federated framework is in the Classification task.

4.1. MNIST Experiment

We first decide to validate our implementation of the federated framework by using the MNIST dataset [25]. The MNIST is a dataset of handwritten digits from 0 to 9. There are 60,000 training images and 10,000 for testing. All the images are centered and normalised.

We use MNIST to be able to compare our performance with [33], and as a result, check the veracity of our implementation. So we simulate the identical settings as is mentioned in [33].

- We distribute the 60,000 images among 100 clients, each client having 600 images.
- During training, we randomly choose only 10% of all clients, which is 10 clients per round.
- One round of communication corresponds to each client computing an update to its copy of the global model based on its local data and communicating this update to the central server.
- We used the same architecture as used in [33] - 2-layer convolutional network.

4.2. Medical Experiment

This thesis aimed to test the feasibility of using FL in the medical domain. So, after the successful experiments on the MNIST dataset [25], we worked towards solving a challenging medical imaging problem.

Medical image classification is a significant real-world problem where the aim is to classify images into respective categories which may help medical practitioners in research or diagnosis.

Now, in a real-world scenario, medical data is distributed among hospitals, and it is not always possible to aggregate it together. So, we decided to apply FL to the task of medical image classification.

4.2.1. Methodology

Medical datasets usually suffer from a critical problem of class imbalance, where the number of instances of a category significantly outnumbers the instances of another category. We use the Dermofit dataset, which also has a severe class imbalance. For example, from Section 4.2.2 we can see the class PYO has only 24 images, whereas ML has 331 images. This problem can lead to sub-optimal performance as the network will be biased towards the over-represented classes and won't have the chance to learn the distributions of the under-represented ones. To tackle this problem, we need to weight each class according to the number of instances in that class. These weights are obtained via Median Frequency Balancing [12], which we explain below.

Let the 10 classes $(1, 2, ..., 10)$ of Dermofit have $N_1, N_2, ...N_10$ images. We compute the median of the number of images of the 10 classes and term it N_{median}.
Then for a class c, we define its weight by equation 4.1.

$$W_c = \frac{N_{median}}{N_c} \tag{4.1}$$

These weights $(W_1, W_2, .., W_10)$ are used in the loss function during the training. The loss function we use for this task is Weighted Cross Entropy. For a multi-class classification task, the weighted cross entropy is defined by,

$$L = -\sum_i W_i y_i \log(p_i) \tag{4.2}$$

Here in equation 4.2, i refers to the class, W_i is the weight calculated using equation 4.1, y_i is the true label and p_i is the prediction of the model.

For our experiments, we divide the dataset into a training set with 653 images and a test set with 647 images. This ensures that both the training and testing set has enough images belonging to each class. Then, we distribute the training set among the clients equally and randomly to replicate a realistic, federated setting.

We choose Resnet-18 [15], pre-trained on Imagenet [8] as our network for all experiments. The architecture is shown in Fig. 4.2.

To compare our results, we use average accuracy and confusion matrix as our metrics.

4.2.2. Dataset

The dataset we use for our medical experiments is called Dermofit [3]. It is a dataset of skin lesions. The images are normal RGB taken with an SLR camera under controlled artificial lighting. There are 1300 images in total, distributed into 10 different classes (cf. 4.1). The different types are Actinic Keratosis (AK): 45, Basal Cell Carcinoma (BCC): 239, Melanocytic Nevus / Mole (ML): 331, Squamous Cell Carcinoma (SCC) sample 88, Seborrhoeic Keratosis (SK): 257, Intraepithelial carcinoma (IEC): 78, Pyogenic Granuloma (PYO): 24, Haemangioma (VASC): 96, Dermatofibroma (DF): 65, Melanoma (MEL): 76.

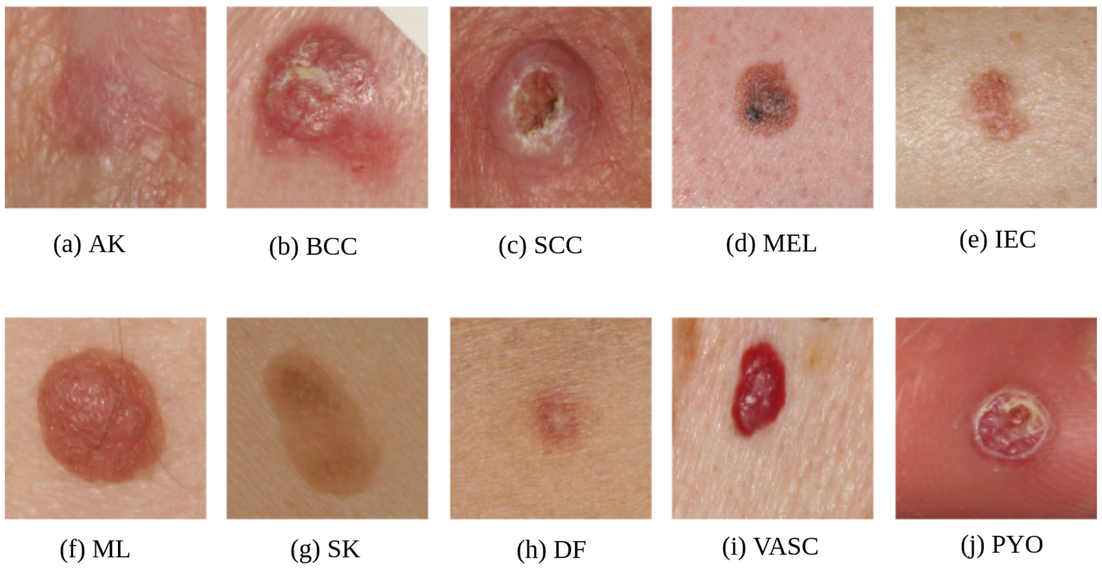


Figure 4.1.: Sample images from each class of Dermofit, showcasing the variability within the dataset [3]

4.2.3. Experimental Setup

We conduct two types of experiment, 1) effect of local computation and 2) effect of the number of clients. For all our experiments, we use a learning rate of 1e-4 with Adam optimizer [20].
We used Pytorch framework [37] to implement our code and trained the models on one NVIDIA GPU Titan XP.

4.2.4. Baseline

We chose a centrally (all the data in one place) trained pre-trained Resnet-18 [15] as our baseline, which reached a maximum accuracy of 83%.

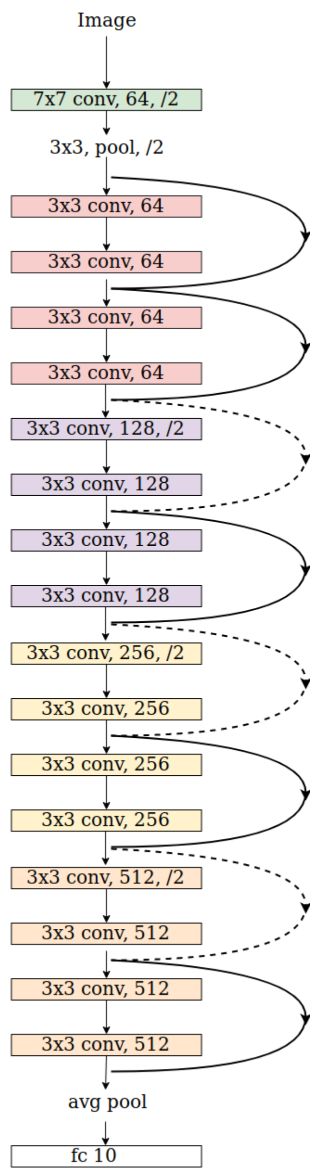


Figure 4.2.: Resnet-18 Architecture with 17 convolutional layers with skip connections after every two layers, ending with the final fully connected classifier block. [15]

5. Federated Segmentation

Medical Image Segmentation is the process of detecting a specific object or the outline in a 2D or 3D image. Segmentation can be automatic or guided by the user. A crucial problem of medical image segmentation is the high variability in medical images. Human anatomy can vary widely among different subjects. Also, there exist various imaging modalities (MRI, X-ray and CT) which differ in the level of detail and resolution achieved in the scans. Segmentation results can be utilised to obtain further diagnostic insights. Some common applications can be the automatic measurement of organs, cell counting, and detection of tumors or malignancy.

Learning to segment medical image is a challenging task, but it has huge positive repercussions. It is, therefore, at the forefront of medical computing research around the world. Now, as is with most medical imaging tasks, the major bottleneck to building an efficient segmentation model is data. CT or MRI scans come in the form of volumes or 3D images. These scans are expensive to obtain and are bound by privacy regulations. So, there is a scarcity of free or open datasets of such images.
This problem can be solved by using Federated Learning. For our report, we implement a federated segmentation framework that performs whole-brain segmentation.

5.1. Whole-Brain Segmentation

Brain segmentation is an important quantitative method in medical imaging, which provides a non-invasive way to analyse the different regions of a brain. Whole-brain segmentation can be used in a longitudinal study to monitor changes in the size and shape of brain structures. For example, variations in brain anatomies, such as the Hippocampus can indicate the progression of diseases like Alzheimer's. Magnetic Resonance Imaging (MRI) is generally used for whole-brain segmentation. A neural network is trained with MRI scans to classify tissues in and around the brain.
In our experiments, the neural network learns to classify each pixel of the input image into its corresponding true class. By this method, we can detect regions of pixels with the same class. This process is called semantic segmentation.

5.1.1. Methodology

To simulate FL, we had to distribute the dataset among many clients. In a realistic setting, the volume of one patient will remain with one client or hospital. So, we needed to make

sure that image slices from the volume of a patient were not distributed among different clients.
So, before splitting the dataset into slices, we distributed the volumes of the patient among the clients and then locally pre-processed each client dataset. This ensured that the data of a single patient always remained with the same client.

For the segmentation network, we use QuickNAT [40]. It is a fully convolutional, densely connected network that can produce accurate segmentation results very quickly. QuickNAT uses Squeeze and Excitation blocks [16, 42]. These blocks help to tackle the problem of lack of pre-trained models for medical data by generating a task-specific representation which helps in final segmentation. This improves the performance and generalizability of the model.

To train the network, we implement a combination of dice and cross-entropy loss from [48].

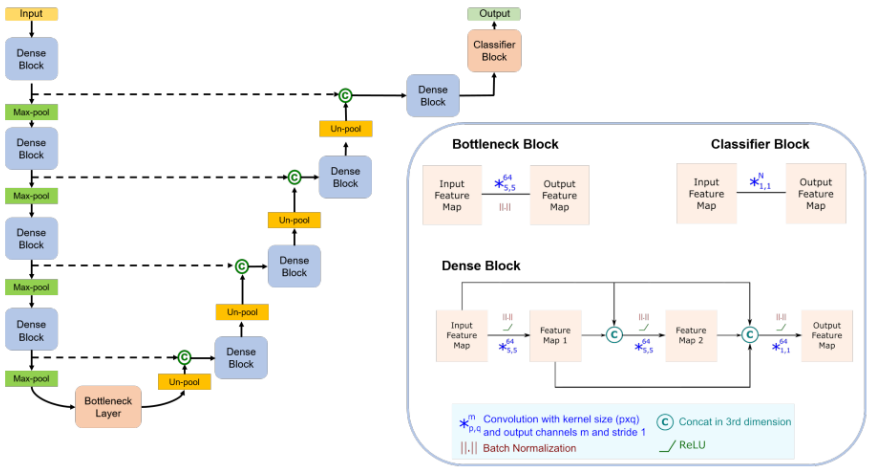


Figure 5.1.: QuickNAT containing detailed architecture of the dense block, classifier block and the bottleneck layer. [40]

To compare the experiments, we use Dice Coefficient (DC) as our metric. It is a similarity measure in the range [0,1] which provides the ratio of intersection over union of the predictions (P) and true labels (T). It is defined as:

$$DC = \frac{2|P \cap T|}{|P| + |T|} \tag{5.1}$$

5.1.2. Dataset

We use the Multi-Atlas Labelling Challenge (MALC) [31] dataset for our whole brain segmentation experiment. MALC is part of the Open Access Series of Imaging Studies (OASIS) [31] dataset. The OASIS is a multi-modal dataset, aimed at facilitating future research

on computational neuroscience and brain segmentation.
MALC contains 30 whole-brain MRI T1 scans with manual annotations from Neuromorphometrics, Inc. [53]. The input volumes are sized 256×256×256. We used 2D slices of 256×256 images for our network. We extracted the slices from the volumes in coronal orientation. We used 20 volumes for training and 10 for testing.
We considered 33 classes for the segmentation, following [40], using the FS label remapping function that was used in [40].

5.1.3. Baseline

We take a centrally trained segmentation model as our performance upper bound. This refers to the ideal scenario when all the data are aggregated in one place, and a single model can be trained on it. Additionally, there is no privacy aspect to take care of. Our baseline achieves best DC of 0.867.

5.1.4. Experimental Setup

We conduct two types of experiment, 1) effect of local epochs and 2) effect of number of clients. We use the recommended network settings as stated in [40]. We use Adam [20] optimizer and learning rate of 1e-4 for all our experiments.
We implemented our code in the Pytorch framework [37] and trained the models on one NVIDIA GPU Titan XP.

5.2. Brain Segmentation with Loss Balancing

The simple FL approach performs well for fewer clients, but for a highly distributed task which has many clients, FL does not converge well. Sheller et al.[46] showed that for segmentation, with an increase in the number of clients, the performance decreased. So we theorised a possible solution - to modify the averaging algorithm of the local models.

5.2.1. Methodology

We make use of the concept called Median Frequency Balancing [12]. We speculate that a client whose loss is lower has achieved better convergence and so its model parameters are more useful for the global model. So, we try to weight each client according to their importance to the global model, based on their respective losses. We use the client losses per communication round to calculate weights for the corresponding client, which are then applied to their model parameters during averaging.

Let us assume, following the first communication round t, a client C_k has a training loss L_k^t and its trained model parameters are P_k^t. It returns all these variables to the global server. Then after all clients been trained, we compute the median of all client losses,

L^t_{median}, at the server-side.
Then for a client k, at communication round t, we define its weight by equation 5.2.

$$W^t_k = \frac{L^t_{median}}{L^t_k} \tag{5.2}$$

After we calculate the weights for every client, we multiply them with their model parameters to weigh them according to their importance, as per the equation 5.3. The updated parameters are then averaged together to get the global mode.

$$P^t_k = P^t_k \times W^t_k \tag{5.3}$$

5.3. Brain Segmentation from Multiple Dataset

In the real world, hospitals have different scanners and MRI modalities for their scanning procedure. It is therefore, vital to see whether FL works with different datasets in each client. In this experiment, we make use of two datasets, namely, MALC and IBSR [7].

5.3.1. Datasets

We introduce a new dataset for this experiment. The Internet Brain Segmentation Repository (IBSR) dataset [7] provides manually segmented labels along with the magnetic resonance brain image data. The IBSR dataset consists of 18 real MRI image volumes derived from healthy subjects. Each MRI volume has a size of 256×256×128. For our experiment, all images have segmented labels of 33 classes in the brain. This is kept similar to the MALC dataset. We take 16 volumes for our training and distribute them among the clients. We keep 2 volumes for testing.
The IBSR has a much lower resolution than MALC, which makes the problem harder, as the network needs to learn to segment input of two different resolutions.

5.3.2. Experiment Setup

In all experiments, we distributed the IBSR training set between 2 new clients, with each client containing 8 MR volumes. For validation, we keep 10 volumes from MALC and 2 volumes from IBSR. The other settings were kept the same as described in Section 5.1. We do not use loss balancing for this experiment.
We used Pytorch framework [37] to implement our code and trained the models on one NVIDIA GPU Titan XP.

Part III.

Federated Experiments and Results

6. Federated Classification

6.1. MNIST Results

We compared the results from our experiment to the findings of [33]. The following table compares the number of communication rounds needed to reach an accuracy of 99%.

Local Epochs	Batch Size	Comm Rounds [33]	Comm Rounds Ours
1	∞	626	669
5	∞	179	349
20	∞	234	114
1	10	34	346
5	10	20	66
20	10	18	43

Table 6.1.: Communication rounds needed to reach 99% test accuracy on MNIST for different combinations of local epoch and local batch size. Comparison of [33] and our implementation.

In Table 6.1, we try out different local epochs and batch sizes. ∞ batch size means training the network with the entire client dataset at once, without splitting it into mini-batches. As seen in Table 6.1, we were able to reproduce the findings of [33] using our implementation of the federated framework. Our results show a trend, that is, with a smaller client batch size, the global model converges faster and requires fewer communication rounds. This results in lower communication costs and training time. The authors of [33] also saw this trend. The cause of this trend can be attributed to the fact that with a larger batch size, the model overfits on the local datasets and so the global model is unable to generalize well.

Furthermore, it can be observed from our results that by increasing local epochs, we decrease the number of communication rounds needed to achieve the best accuracy. This pattern is uniform for any batch size. The reason behind this pattern can be attributed to the fact that each local model converges faster with higher local epochs and so, the global model is able to reach the best performance. Thus, it is always advisable to increase local computation. However, it is important to keep in mind that with too many local epochs, the local models may overfit which will prevent the global model from converging. It is therefore essential to find the optimal value.

6.2. Medical Dataset Results

6.2.1. Effect of Local Computation in a Client

Local Epochs	Batch Size	Max Accuracy (%)	Comm Rounds	Time (minutes)
1	∞	78.00	163	55
5	10	82.00	20	19
5	∞	81.00	40	44
10	10	82.00	13	22
10	∞	81.00	40	77
15	**10**	**82.50**	**19**	**54**
15	∞	81.00	12	33
20	10	81.00	11	34
20	∞	80.00	17	62

Table 6.2.: Effect of local epoch and batch size on the accuracy of the Dermofit dataset.

In the first experiment, we distribute the dataset among 5 users, with each user having about 130 images. Our aim is to find the best combination of local epoch and client batch size for our framework. We aim to achieve accuracy as close to the baseline while keeping the communication costs and time low. Table 6.2 shows the various combinations of batch size and local epochs that we tried. ∞ batch size means using the entire client dataset for a iteration over the network.
The first row, with batch size ∞ and epoch of 1 corresponds to the naive FedSGD mentioned in [33]. As we can see this simple version does not perform well. We were only able to achieve the best accuracy of 78%, which is much below the baseline. It took 163 communication rounds and 55 minutes to achieve this performance.
Next, we tried FedAvg [33] where we increase the computations per client by increasing the local epochs and decreasing the batch size for each client. It is evident that by increasing client computation, we achieve superior results. When we have less client computation, the local models were not trained enough and so underfitted on the data. By increasing computation, the local models converge better. and so the global performance is also improved.
Our best performance with 5 clients- **Accuracy of 82.50 with 15 local epochs, 10 batch size, and 19 communication rounds**.

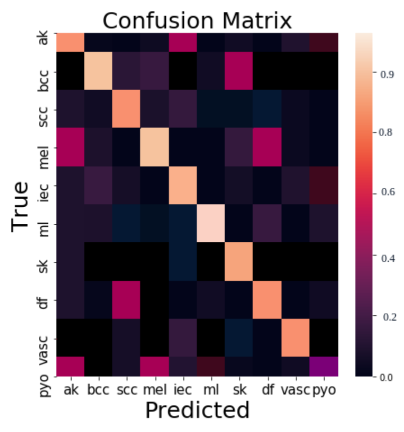
6.2.2. Effect of Number of Clients

We conducted a second experiment to see how increasing the number of clients can affect our framework. As seen in Table 6.2, our best result was achieved by using a batch size of 10 and 15 local epochs. So, we set the batch size and local epochs to the optimal values of

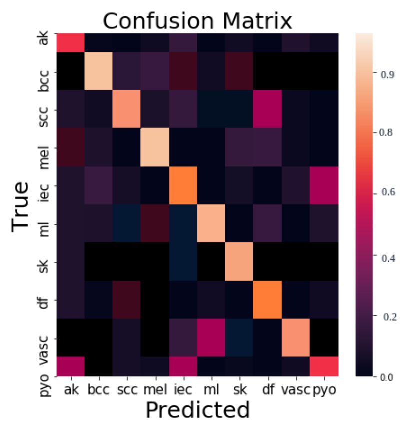
Number of Clients	Maximum Accuracy (%)	Communication Rounds
1	83.00	-
2	**83.00**	**13**
5	82.50	19
10	82.00	6
15	81.00	15
20	81.00	16

Table 6.3.: Effect of number of clients on the performance of our model on Dermofit.

10 and 15 respectively. The first row in Table 6.2 is the baseline we compare our method with.
We distribute or training set among 2, 5, 10, 15 and 20 clients, with each client dataset having 325, 130, 65 and 32 images respectively. Table 6.3 show our findings, which indicate that a highly distributed dataset adversely affects the performance of a network. With an increase in clients, the accuracy of the network decreases. There can be two reasons behind it.
When we increase clients, the size of the client dataset becomes smaller, which can lead to the overfitting of the local models on local data. Then the global model does not learn to generalise.
Another reason for this result could be that the federated framework cannot handle a very high number of clients. Averaging the weights from too many different client models may not provide the desired global model.



Baseline Result



Our Result

Figure 6.1.: Confusion Matrices showing the class wise accuracy of our model on the Dermofit testset compared to the baseline performance.

In Fig. 6.1, we plot heatmaps of the confusion matrix of our best performing model on the Dermofit dataset and the baseline model. A confusion matrix plots the correctly classified samples in the diagonal, and all the miss-classified examples in the non-diagonal. It helps to understand the performance of a model for each individual class.
We see that our model has a low miss-classification error and is able to achieve results comparable to the baseline model. There are classes like PYO and AK which have a lower accuracy. This is because these classes are underrepresented in the dataset. However, our model is still able to achieve good performance uniformly across the classes.

Our experiments show that our federated network can match the performance of a centrally trained network for the task of classification. We demonstrate this using a conventional vision dataset (MNIST) and also with a medical imaging dataset (Dermofit).
Our federated model successfully overcomes the challenges that are associated with medical datasets, namely, lack of data and no pre-trained models. It achieves performance on par with our baseline while preserving the privacy of each client dataset.

7. Federated Segmentation

7.1. Whole-Brain Segmentation Results

7.1.1. Effect of Local Training Epochs on Performance

For our first experiment, we try different combinations of local epochs. We fix the number of clients for this experiment to 4. So we distribute 20 training volumes among 4 clients, each client having 5 volumes. We train each time for 30 communication rounds (global epoch). Table 7.1 shows the results of our experiments.

Number of Clients	Local Epochs	Communication Rounds	Best Dice Score
4	1	30	0.746
4	**2**	**30**	**0.815**
4	5	30	0.794
4	10	30	0.789
4	15	30	0.768

Table 7.1.: Effect of local epochs on federated segmentation

From Table 7.1 we notice a distinct trend. For the task of segmentation, as we increase the local epochs, the best dice decreases. This can be explained by the fact that each client has only 5 patient volumes. So, by increasing local computation, we risk overfitting on the local dataset. This hinders the convergence of the global averaged model.
On the other hand, with a local epoch of 1, we also get poor performance. This can be attributed to inadequate training of each client.
So, a middle ground of 2 local epochs yields the best result.

7.1.2. Effect of Number of Clients on Performance

The first experiment helped to understand how local epoch affected training. From the results, we saw that 2 local epochs are optimal for our task.
The second experiment we conducted was to see how the number of clients can affect the training of the network. So, we trained our network for various number of clients and recorded the results. We kept the local epochs fixed at 2 and ran all experiments for 50 communication rounds. The Table 7.2 shows our results. The first row corresponds to our baseline, which is the centrally trained segmentation network.

Number of Clients	Volumes per Client	Local Epochs	Comm Rounds	Best Dice
1	20	50	-	0.867
2	**10**	**2**	**50**	**0.847**
4	5	2	50	0.838
5	4	2	50	0.823
10	2	2	50	0.816
20	1	2	50	0.773

Table 7.2.: Effect of number of clients on federated segmentation

Increasing the number of clients leads to a decline in the performance of the model. Two factors can explain this.
Firstly, we have only 20 volumes for training which we distribute among the clients. Now, by increasing the number of clients, we effectively reduce the size of the dataset each client has. For example, when we have 20 clients, each client has 1 volume, which it uses to train the network. This is not enough, and hence the performance suffers.
The second factor could be the weighted averaging. When we average too many local models, we lose crucial features learnt by them. The averaged model forgets the feature representations learnt by the individual client model from its data, especially in the case of many clients. So, the global average model cannot represent the client datasets accurately.

From this experiment, we learnt that having too many clients can adversely impact the performance of the federated segmentation model. One solution would be to add more data per client. However, we do not have a bigger dataset, so this option is out of the scope of this thesis. So, in our next section, we implement the Loss Balancing introduced in Section 5.2 to counter this issue.

7.2. Whole Brain Segmentation with Loss Balancing

From Experiment 7.1.2, we identified the problems associated with having many clients. To find the cause of this problem we analysed the loss curves. We found that for a larger number of clients, there are sudden fluctuations in the training curves (c.f 7.1 (A)). We assume this is due to poor averaging of all the client models.

After incorporating median loss balancing, we see an increase in the performance of the network in case of a greater number of clients. The table 7.3 depicts our findings.
We also plot the loss curves where we can see our solution has successfully addressed the problem of loss fluctuations, as can be seen in Figure 7.1 (B).

This experiment improves upon the results from Table 7.2 for the higher number of clients. Also, for fewer clients, the performance is not majorly affected.

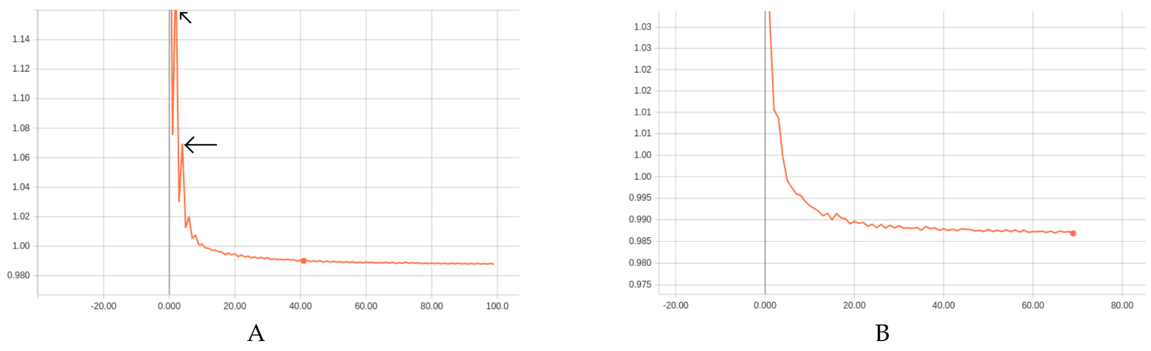


Figure 7.1.: Training loss (A) for 10 clients using simple FL has many jagged peaks. The loss curves (B) becomes smoother and is devoid of any sharp peaks, after implementing Loss Balancing.

No. of Clients	Vol. per Client	Comm. Rounds	DC with FL	DC with Balancing
2	**10**	**50**	**0.847**	**0.842**
4	5	50	0.838	0.840
5	4	50	0.823	0.832
10	2	50	0.816	0.831
20	1	50	0.773	0.791

Table 7.3.: Loss Balancing Results

7.3. Comparison of FL with other Distributed Methods

In this section, we compare our method of federated segmentation with the baseline, BrainTorrent [41] and two other distributed machine learning methods which are defined below.

We use the same model architecture, QuickNAT [40] and the entire MALC [31] training set for the centralised training.

Institutional Incremental Learning (IIL)

IIL is a distributed learning method, where the clients train a shared global model one after another. Each client is restricted to training the model once. However, they have the freedom to choose the training process, as long as the global model architecture is not altered. IIL incurs less communication cost, as the clients need to transmit the model once.

IIL suffers from two main drawbacks:

- Increasing the number of clients leads to a drop in performance.
- Features learned previously are forgotten and replaced with the information learnt from new data. This is called catastrophic forgetting [13].

Cyclic Institutional Incremental Learning (CIIL)

CIIL is a modified form of IIL, where we repeat the IIL process multiple times in a cyclic manner. Also, each client is restricted to training a fixed number of epochs. This prevents the problem of catastrophic forgetting [13].
CIIL has one flaw; it needs additional infrastructure to include a validation process. This makes CIIL much more complex and inefficient.

We compare the results from these methods in table 7.4. For the distributed learning methods, we distribute the data among four clients. All methods have the same network architecture QuickNAT [40], Adam optimizer [20] and a learning rate of 1e-4.

Type of Method	Validation Best Dice Score
Baseline	0.867
FL with Loss Balancing	**0.840**
Brain Torrent [41]	0.863
CIIL	0.829
IIL	0.792

Table 7.4.: Quantitative Comparison of the Methods.

Federated Learning outperforms CIIL and IIL. It cannot surpass the baseline of centralised training. Nevertheless, this is anticipated since by averaging multiple models, we lose some features learnt from each client model. BrainTorrent [41] also has a superior performance compared to our method, but it cannot ensure privacy as all clients communicate with each other directly. So our model can match the performance of other methods while ensuring that the privacy of each client dataset is not compromised.

7.3.1. Qualitative Comparison

It is always easier to analyse results visually, and it also helps to understand better how the network is performing. Here, we compare the outputs of the various methods mentioned in Section 7.3. Each distributed method has 4 clients and the same architecture backbone.

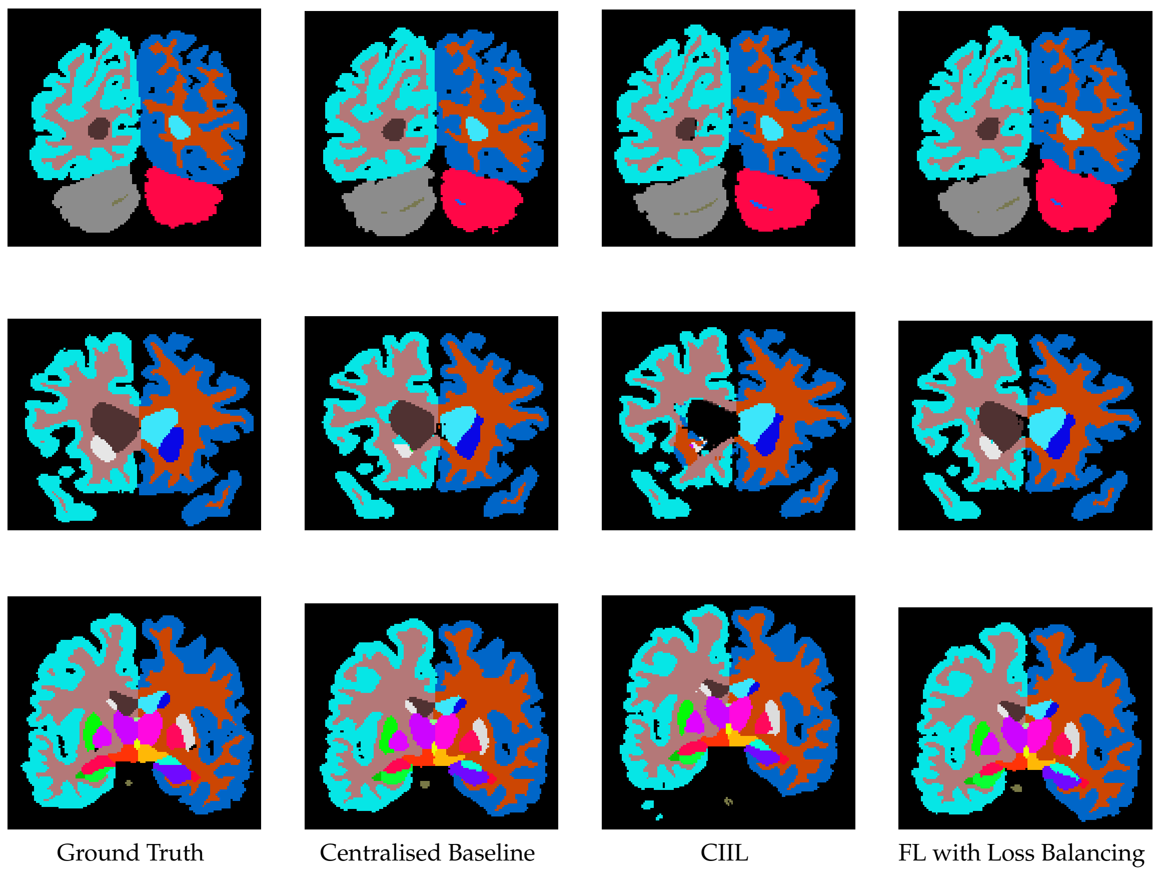


Figure 7.2.: Qualitative Comparison of Various Segmentation Methods.

We see that qualitatively also, our methods perform as good as the centralised baseline. The CIIL fails to segment the input in the second row, but our method has no trouble with it.

7.4. Segmentation of Multiple Dataset Results

The results of the experiments defined in Section 5.3 are shown in Table 7.5.

In Table 7.5, we see that adding another dataset lowers the overall performance from Table 7.3. This can be due to the different properties of the two datasets. The IBSR images have lower resolution compared to MALC. So, the network had to learn to take resolution into account when segmenting the inputs. This made the problem much more complex and therefore, harder to solve. We conclude that there is scope for further research in this direction.

MALC Clients	IBSR Clients	Local Epochs	Comm Rounds	Best Dice
2	**2**	**2**	**50**	**0.827**
4	2	2	50	0.811
5	2	2	50	0.810
10	2	2	50	0.791

Table 7.5.: Segmentation with Multiple Datasets

7.4.1. Qualitative Results

We also plot the predicted segmentation results from the testset comprising of both MALC and IBSR. We see that our model generalizes well and is able to learn the different features of both datasets successfully. The IBSR results are slightly worse than the MALC results. This can be since we had fewer IBSR images than MALC during training. Also, IBSR scans are of poor quality, and that could have led to worse results.

Figure 7.3.: IBSR Segmentation Results

Figure 7.4.: MALC Segmentation Results

Part IV.

Conclusion

8. Medical Applications of Federated Learning

FL has been widely used in smart devices and smartphones. It takes advantage of the highly distributed data present in these devices to learn machine learning models while keeping the privacy of the data of each device intact.
This idea of FL can be extremely useful in the medical domain. In this work, we applied the concept of federated learning to solve medical problems. Our experiments show the feasibility of using federated learning for the tasks of Classification and Segmentation, using medical data obtained from hospitals and medical institutions.

8.1. Federated Classification

Classification is a common task in the medical field. One such application is the classification of skin lesion images. This task involves classifying natural images of lesions into their correct category.
We implemented federated classification of skin lesions, where the dataset Dermofit [3] was divided among clients equally. Our federated model achieved an accuracy of 82.5%, which is very close to our baseline accuracy of 83% by the model trained with aggregated data. Our model matches the performance of centralised training while conserving the privacy of data of each client.

8.2. Federated Segmentation

Segmentation is an important task in the field of medical imaging. Its application range from organ segmentation to tumor detection. One of the most popular tasks nowadays is whole-brain segmentation. In this report, we apply FL to the task of whole-brain segmentation. The dataset MALC [31] was divided among several clients to simulate a distributed scenario. We experimented with a various number of clients to analyse how a highly distributed setting affects the performance. We also introduced a solution of Loss Balancing, to counter the problem of poor performance when there are a high number of clients. Our solution provided stability to the training process, which resulted in better performance. Our federated model for two clients has the best dice of 0.847, which is comparable to the baseline model with a dice of 0.867. We also compared our method with distributed learning strategies like IIL and CIIL. Our experiments showed that these methods perform worse than FL and also do not scale well with larger number of clients. Lastly, we also conducted a federated experiment with multiple datasets on different clients. Our model was able to generalise well from two different datasets.

8.3. Future Work

From our experiments, it is proved that FL can be applied to any medical application without any serious loss in performance, all while maintaining the privacy aspect of the medical data.
We show that having a large number of clients affects the training adversely leading to poorer performance. We propose Loss Balancing, which improves the performance for highly distributed cases. However, more research is needed in this area.
Further work can be done in the privacy aspect of federated learning by incorporating differential privacy and asynchronous encryption to the framework. Another very unique area of research that has huge potential and can have a positive impact on the medical community is federated multi-task learning.

We show that it is beneficial to adopt FL in clinical applications as it enables the assimilation of knowledge from different sources for learning a single model. This process improves the performance of the global model without the need to share medical data, thereby overcoming any privacy or data ownership concerns.

Appendix

A. Additional Segmentation Results

A.1. Detailed Ablation Study

No. of clients	Vol per Client	Local Epochs	Comm Rounds	Best Dice Score
2	10	1	50	0.838
2	**10**	**2**	**50**	**0.847**
2	10	3	50	0.824
2	10	5	50	0.823
2	10	10	50	0.818
4	5	1	30	0.746
4	**5**	**2**	**30**	**0.838**
4	5	3	30	0.819
4	5	5	30	0.803
4	5	10	30	0.789
4	5	15	30	0.767
5	4	1	30	0.727
5	**4**	**2**	**30**	**0.823**
5	4	5	30	0.782
5	4	10	30	0.754
10	**2**	**2**	**30**	**0.816**
10	2	5	30	0.740
20	**1**	**2**	**30**	**0.773**
20	1	5	30	0.674

Table A.1.: Federated whole-brain segmentation ablation study of various number of clients and various local epochs.

A.2. Other Distributed Segmentation Methods

No. of Clients	Local Epochs	Comm Rounds	Best Dice
2	**2**	**50**	**0.805**
4	2	50	0.792
10	2	50	0.784

Table A.2.: IIL Ablation Study

No. of Clients	Local Epochs	Comm Rounds	Best Dice
2	**2**	**50**	**0.830**
4	2	50	0.829
10	2	50	0.820

Table A.3.: CIIL Ablation Study

List of Figures

List of Tables

Bibliography

[1] Sarah S Aboutalib, Aly A Mohamed, Wendie A Berg, Margarita L Zuley, Jules H Sumkin, and Shandong Wu. Deep learning to distinguish recalled but benign mammography images in breast cancer screening. *Clinical Cancer Research*, 24(23):5902–5909, 2018.

[2] Marwan Ali Albahar. Skin lesion classification using convolutional neural network with novel regularizer. *IEEE Access*, 7:38306–38313, 2019.

[3] Lucia Ballerini, Robert B Fisher, Ben Aldridge, and Jonathan Rees. A color and texture based hierarchical k-nn approach to the classification of non-melanoma skin lesions. In *Color Medical Image Analysis*, pages 63–86. Springer, 2013.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[5] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[6] Darvin Yi Carson Lam, Margaret Guo, and Tony Lindsey. Automated detection of diabetic retinopathy using deep learning. *AMIA Summits on Translational Science Proceedings*, 2018:147, 2018.

[7] Chris A Cocosco, Vasken Kollokian, Remi K-S Kwan, G Bruce Pike, and Alan C Evans. Brainweb: Online interface to a 3d mri simulated brain database. In *NeuroImage*. Citeseer, 1997.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] Amol Deshpande, Carlos Guestrin, Samuel R Madden, Joseph M Hellerstein, and Wei Hong. Model-based approximate querying in sensor networks. *The VLDB journal*, 14(4):417–443, 2005.

[10] M Murat Dundar, Glenn Fung, Balaji Krishnapuram, and R Bharat Rao. Multiple-instance learning algorithms for computer-aided detection. *IEEE Transactions on Biomedical Engineering*, 55(3):1015–1021, 2008.

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[12] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[13] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[14] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[17] Li Huang, Andrew L Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of Biomedical Informatics*, 99:103291, 2019.

[18] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. Loadaboost: Loss-based adaboost federated machine learning on medical data. *arXiv preprint arXiv:1811.12629*, 2018.

[19] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] Tsvi Kuflik, Judy Kay, and Bob Kummerfeld. Challenges and solutions of ubiquitous user modeling. In *Ubiquitous display environments*, pages 7–30. Springer, 2012.

[23] B Landman and S Warfield. Miccai 2012 workshop on multi-atlas labeling. In *Medical image computing and computer assisted intervention conference*, 2012.

[24] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differentially private meta-learning. *arXiv preprint arXiv:1909.05830*, 2019.

[27] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

[28] Tian Li, Maziar Sanjabi, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019.

[29] Wenqi Li, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M Jorge Cardoso, et al. Privacy-preserving federated brain tumour segmentation. In *International Workshop on Machine Learning in Medical Imaging*, pages 133–141. Springer, 2019.

[30] Dianbo Liu, Timothy Miller, Raheel Sayeed, and Kenneth Mandl. Fadl: Federated-autonomous deep learning for distributed electronic health record. *arXiv preprint arXiv:1811.11400*, 2018.

[31] Daniel S Marcus, Tracy H Wang, Jamie Parker, John G Csernansky, John C Morris, and Randy L Buckner. Open access series of imaging studies (oasis): cross-sectional mri data in young, middle aged, nondemented, and demented older adults. *Journal of cognitive neuroscience*, 19(9):1498–1507, 2007.

[32] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data, Apr 2017.

[33] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[34] Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2014.

[35] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*, 2019.

[36] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[38] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019.

[39] Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5, 2018.

[40] Abhijit Guha Roy, Sailesh Conjeti, Nassir Navab, Christian Wachinger, Alzheimer's Disease Neuroimaging Initiative, et al. Quicknat: A fully convolutional network for quick and accurate segmentation of neuroanatomy. *NeuroImage*, 186:713–727, 2019.

[41] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[42] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. 'squeeze & excite'guided few-shot segmentation of volumetric images. *Medical image analysis*, 59:101587, 2020.

[43] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[44] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[45] U Joseph Schoepf, Alex C Schneider, Marco Das, Susan A Wood, Jugesh I Cheema, and Philip Costello. Pulmonary embolism: computer-aided detection at multidetector row spiral computed tomography. *Journal of thoracic imaging*, 22(4):319–323, 2007.

[46] Micah J Sheller, G Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. Multi-institutional deep learning modeling without sharing patient data: A

feasibility study on brain tumor segmentation. In *International MICCAI Brainlesion Workshop*, pages 92–104. Springer, 2018.

[47] Li Shen, Laurie R Margolies, Joseph H Rothstein, Eugene Fluder, Russell McBride, and Weiva Sieh. Deep learning to improve breast cancer detection on screening mammography. *Scientific reports*, 9(1):1–12, 2019.

[48] Shayan Ahmad Siddiqui and Abhijit Guha Roy. Nn common modules.

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[50] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[52] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[53] Andrew Worth. Neuromorphometrics, inc.

[54] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Probabilistic federated neural matching. 2018.

[55] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.