

THE UNIVERSITY OF HONG KONG
DEPARTMENT OF COMPUTER SCIENCE

FYP 20060 Final Report (Individual)

Federated Learning Platform for Covid-19 Detection

Ghosh Bratin 3035437692 (author)

Khan Khondoker Araf Hasan 3035477446

Kwan Pok Man 3035477173

Under the humble guidance of Prof. S.M. Yiu

April 18, 2021

Abstract

With the modern era of big data, obtaining data to build Machine Learning models is becoming more and more scarce due to security reasons. Hence, switching to a method where users can collaboratively train a global model without risking any compromise of sensitive data is increasingly necessary to gain better insights.

Federated Learning is a new concept that arose in 2016, that hopes to tackle the aforementioned issue. This project focusses on creating a federated learning platform for the detection of COVID-19 virus in patients. This will enable various hospitals to work together to develop an aggregated model without fearing the loss of patient's information while maintaining high precision.

The core part of this project is to research, design and propose a platform where different users can jointly train a machine learning model which can achieve the same performance as that of a traditional machine training method.

The methodology describes the general structure of the platform allowing multiple clients communication with the server to build the Machine Learning model. Research and evaluation on different federated learning methods in Tensorflow federated has been done and a basic client-server communication using Python Flower framework has been developed. Alongside that, a 1-to-1 training platform has been created using Pysyft framework and a platform using Pysyft, Pygrid and Syft.js. The project can be extended in the future which involves deploying the Federated Learning Command Line Interface platform and development and debugging of the Federated Learning web platform using Pysyft, Pygrid and Syft.js.

Acknowledgement

I would like to express my sincere gratitude to everyone who had a role to play in the inception and progress of the FYP project. A special thank you to Professor S. M. Yiu for his critical suggestions, which greatly helped us. Thank you to my group mates, Araf Khan and Owen Kwan, who have worked tirelessly on this project with me, and without them I am sure this project would not been possible. Finally, I would like to extend my warm gratefulness to The Department of Computer Science of The University of Hong Kong (HKU) for giving me the humble opportunity to work on this FYP project.

Contents

<i>Abstract</i>	1
<i>Acknowledgement</i>	2
<i>List of Figures</i>	5
<i>List of Tables</i>	7
<i>List of Abbreviations</i>	8
<i>List of Symbols</i>	9
1 Introduction	10
1.1 Overview of Machine Learning	10
1.2 Federated Learning	11
1.2.1 Three types of Federated Learning	11
1.2.2 Benefits of Federated Learning.....	14
1.2.3 Challenges of Federated Learning	14
1.3 Gboard on Android	15
1.4 Outline of the report	16
2 Objectives and Motivation	17
2.1 Objectives	17
2.2 Motivation	17
3 Methodology	19
3.1 Introduction	19
3.2 Rationale for using Horizontal Federated Learning	19
3.3 Horizontal Federated Learning Architecture	19
3.4 Security Protocol	21
3.4.1 Secure Multi-party Computation	21
3.4.2 Differential Privacy	21
3.4.3 Homomorphic Encryption	22
3.5 Summary	22
4 Experiments and Results	23
4.1 Overview	23
4.2 Tensorflow Federated	23
4.2.1 Fashion MNIST	24

4.2.2	CIFAR10.....	27
4.2.3	Cats_vs_dogs	30
4.2.4	Hyperparameters Comparison.....	32
4.2.4	Limitations	35
4.3	Flower	36
4.3.1	Use Cases.....	37
4.3.2	Rationale behind using Flower	38
4.3.3	CLI Platform and Evaluation.....	38
4.3.4	Deployment.....	41
4.3.5	Difficulties	41
4.4	PySyft, PyGrid and Syft.js.....	42
4.4.1	PySyft.....	42
4.4.1.1	Duet.....	42
4.4.2	PyGrid.....	44
4.4.2.1	Creation of Plans using PySyft	44
4.4.2.2	Hosting on PyGrid using PySyft	45
4.4.2.3	Training using PySyft	45
4.4.3	Syft.js	45
4.4.4	Limitations	46
4.5	Sherpa AI.....	47
5	<i>Difficulties and Limitations</i>	49
5.1	Age of FL Technology	49
5.2	Deployment of FL Platform.....	49
5.3	Complexity of Security Protocols.....	49
5.4	Communication Speed.....	49
6	<i>Future Work</i>	51
7	<i>Conclusion</i>	52
	<i>References</i>	53

List of Figures

- [1.1 Horizontal Federated Learning](#)
- [1.2 Vertical Federated Learning](#)
- [1.3 Federated Transfer Learning](#)
- [1.4 Gboard next word prediction in Gboard](#)
- [3.1 Architecture of Horizontal Federated Learning System](#)
- [4.1 Sample images from Fashion MNIST](#)
- [4.2 Neural network used for Fashion MNIST](#)
- [4.3 Loss and accuracy of neural network trained with traditional machine learning technique on Fashion MNIST](#)
- [4.4 Class distribution of Fashion MNIST](#)
- [4.5 Validation loss and accuracy of neural network trained with Tensorflow Federated Learning API on Fashion MNIST](#)
- [4.6 Validation loss and accuracy of neural network trained with Tensorflow Federated Core on Fashion MNIST](#)
- [4.7 Sample images from CIFAR10](#)
- [4.8 Neural network used for CIFAR10](#)
- [4.9 Loss and accuracy of neural network trained with traditional machine learning technique on CIFAR10](#)
- [4.10 Class distribution of CIFAR10](#)
- [4.11 Validation loss and accuracy of neural network trained with Tensorflow Federated Learning API on CIFAR10](#)
- [4.12 Validation loss and accuracy of neural network trained with Tensorflow Federated Core on CIFAR10](#)
- [4.13 Sample images from Cats vs dogs](#)
- [4.14 Loss and accuracy of neural network trained with traditional machine learning technique on Cats vs dogs](#)
- [4.15 Class distribution of Cats vs dogs](#)
- [4.16 Validation loss and accuracy of neural network trained with Tensorflow Federated Learning API on Cats vs dogs](#)
- [4.17 Validation loss and accuracy of neural network trained with Tensorflow Federated Core on Cats vs dogs](#)
- [4.18 Validation loss and accuracy of neural network trained with the same/different clients for each round](#)
- [4.19 Validation loss and accuracy of neural network trained with different number of clients and number of images per clients](#)

[4.20 Validation loss and accuracy of neural network trained with 5000 images in total](#)

[4.21 Architecture of the Flower Framework](#)

[4.22 Sequence Diagram of one round in the FL CLI platform](#)

[4.23 Frontend design of the web platform](#)

List of Tables

- [4.1 Evaluation Results using the CLI platform \(1\)](#)
- [4.2 Evaluation Results using the CLI platform \(2\)](#)
- [4.3 Evaluation Results using the CLI platform \(3\)](#)
- [4.4 Evaluation Results using the CLI platform \(4\)](#)
- [4.5 Results of simulating multiple clients \(5 epochs\)](#)
- [4.6 Results of simulating multiple clients \(10 epochs\)](#)
- [4.7 Results of simulation with constant number of data](#)
- [4.8 Results of simulation with constant data samples per client](#)

List of Abbreviations

AI	Artificial intelligence
CLI	Command Line Interface
CNN	Convolutional neural network
FCNN	Fully connected neural network
FL	Federated Learning
GAN	Generative Adversarial Network
ML	Machine Learning
SMC	Secure Multi-party Computation
TFF	Tensorflow Federated

List of Symbols

\mathcal{X}	feature space
\mathcal{Y}	label space
I	sample ID space
\mathcal{D}_i	data held by the owner i

1 Introduction

Artificial Intelligence (AI) has been around for more than 50 years. Ever-increasing computational power, big data technologies and ever-improving state-of-the-art algorithms have resulted in major breakthroughs in the field. In recent times, AI's potential has been realised in various industries, including medicine, retail and finance. But AI has been limited by data privacy.

1.1 Overview of Machine Learning

Artificial intelligence is a well-known term in the modern age even among the general public with a limited technology background. However, the definition of AI is constantly evolving, and the term, AI, often gets mangled. Generally speaking, AI describes a machine that can learn from experience and perform a certain task on its own without being explicitly programmed.

Currently, the majority of AI technologies and applications refer to a category known as Machine Learning. ML algorithms incorporate statistics to find patterns given a large piece of dataset. Well known and frequently used technologies such as YouTube recommendation algorithms, Google search engines, Instagram and Facebook feeds are all powered by ML. In the aforementioned instances, each platform is required to collect a substantial amount of data to make accurate predictions on what videos would one like or what post would one most likely react to. These examples underscore the importance of data quantity and quality means in ML. However, the real-world situation is far from ideal.

In recent years, data has become scattered around edge devices (IoT devices such as mobile phones, laptops, tablets, building sensors, drones, security cameras, and wearable health sensors) and advances in technology has exponentially increased the number of edge devices. This has made data ubiquitous, yet hard to obtain. The devices have numerous applications and continuously collect data. But most of this data cannot be shared due to concerns of privacy and can be useful in training machine learning models. In addition, centralized machine learning requires enormous server space storage coupled with world-class security to prevent breaches.

Many countries have shifted their attention to the privacy of data, and hence have formulated strict provisions to regulate data privacy, making it more difficult to obtain data in general. Integration of data is also facing numerous obstacles, and it is difficult to share effectively, leading to the problem of ‘isolated data islands’ in many fields. In order to comply with the privacy laws, the protective measures of data centres can be very costly. The realization of ML in different fields is far more difficult than imagined.

1.2 Federated Learning

In view of such challenges, Google proposed a brand-new concept called "Federated Learning" in 2016 [\[1\]](#). It consisted of a centralized server that coordinates the training activities of clients that are mainly edge devices that use Google’s APIs. In federated learning, unlike traditional machine learning, the data does not need to leave the client side and the client can train the model locally on the edge devices. Connections between the clients and the server are established on the cloud through specific encryption mechanisms, where they share a common global model and that only model updates instead of the data are sent over. The clients will first receive current global model’s weight from the server, then train it with their own local data. After the training is done, the updated parameters, i.e., the local updates, will be sent to the server for aggregation. Through aggregation, the server will consolidate all the updates and produce a more accurate global model. When the clients are idle, they will receive a new global model’s weight from the server and the cycle continues. The aforementioned architecture not only protects the privacy of data of the clients, but also reduces the cost of centralized transmission of large amounts of data.

1.2.1 Three types of Federated Learning

Federated Learning can be classified into horizontal federated learning, vertical federated learning, and federated transfer learning, depending on the type of data.

1.2.1.1 Horizontal Federated Learning

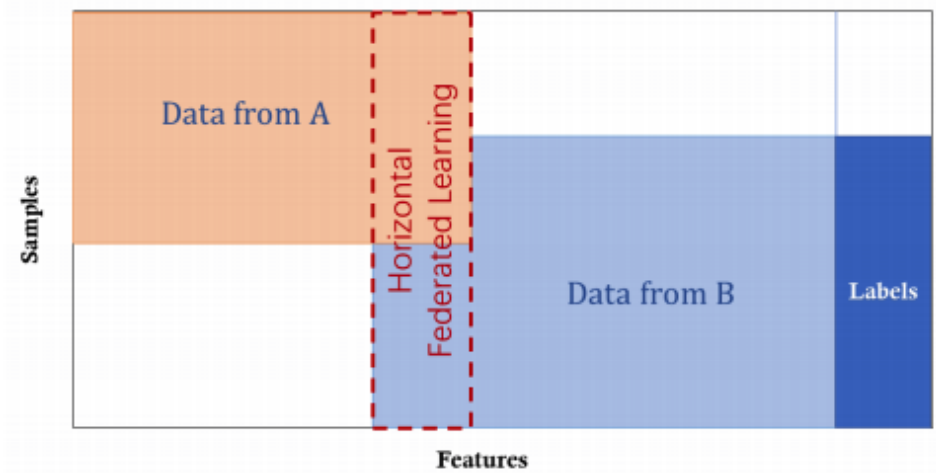


Figure 1.1 Horizontal Federated Learning [2]

Horizontal federated learning is used when participants (clients) have datasets with the same feature space (e.g. sex, height, weight, etc.) but different sample space (e.g. different patients/subjects). First, participants train the model locally, and then send the encrypted model updates to the server. Then the server aggregates the result securely and sends it back to the participants. The participants then update their own model [2].

$$\chi_i = \chi_j, \gamma_i = \gamma_j, I_i \neq I_j, \quad \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j$$

1.2.1.2 Vertical Federated Learning

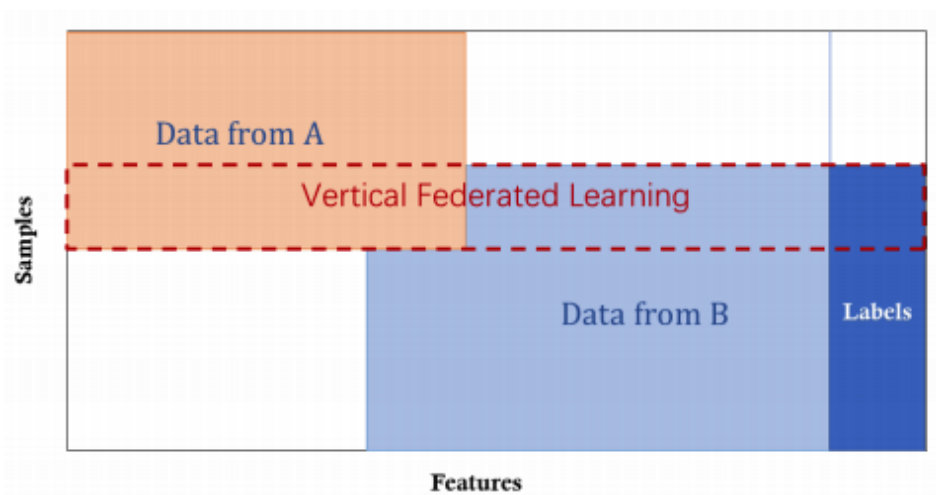


Figure 1.2 Vertical Federated Learning [2]

Vertical federated learning is used when participants have datasets with overlapping sample space but different feature space. In this case, a trusted third party collaborator is involved. The participants first need to confirm the common and overlapping samples. Then the collaborator creates encryption pairs and sends public keys to the participants. Participants encrypt and exchange the intermediate gradient and loss results, compute encrypted gradients and add additional masks, and send the encrypted values back to collaborator. The collaborator decrypts and sends the decrypted gradients and loss back to participants, and participants update their model accordingly [2].

$$\chi_i \neq \chi_j, \gamma_i \neq \gamma_j, I_i = I_j, \quad \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j$$

1.2.1.3 Federated Transfer Learning

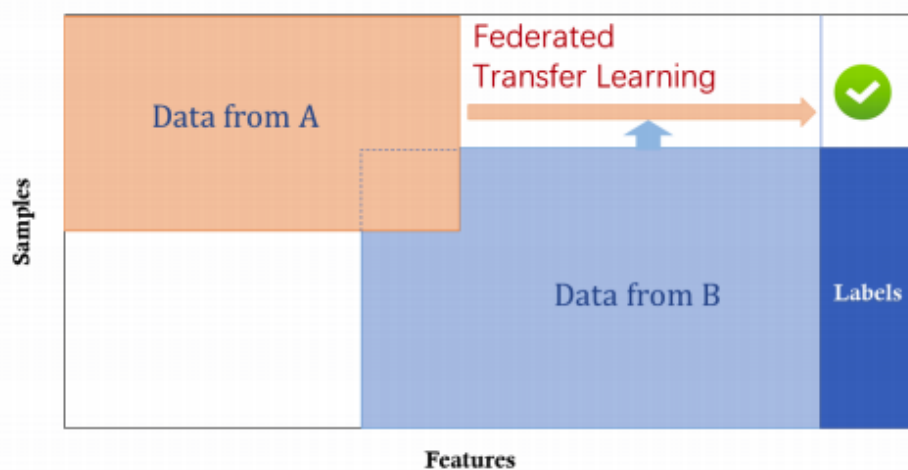


Figure 1.3 Federated Transfer Learning [2]

Federated transfer learning is used when participants have datasets that are different in both feature and sample space. The process is very similar to vertical federated learning, but the gradient computation and intermediate results exchanged between the participants are different [2]. For example, consider a retail company and bank located in two different countries, say India and China. Due to the difference in the geographical location, the users of both companies are bound to be different and hence the intersection of the user group will be very small meaning the sample space is different. Also, both companies operate on completely different business models and contain different type of information about the user meaning the feature space is not similar too. In this situation, transfer learning techniques need to be leveraged in order to allow the sample space and feature space to be used under FL [2]. A common representation needs to be

established using the feature space and the limited sample space. This can later be applied for prediction of samples with one-side features. Federated Transfer Learning is considered as an extension that is important to the existing FL systems since it handles scope beyond the existing FL system.

$$\chi_i \neq \chi_j, \gamma_i \neq \gamma_j, I_i \neq I_j, \quad \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j$$

1.2.2 Benefits of Federated Learning

In FL, the model being trained is present in the device itself enabling real-time operations. The entire dataset need not be transmitted to a central server which vastly reduces the time and cost of data transmission. The edge devices train the model locally and only send the updated weights of the model back to the server to be aggregated into the shared model. The model updates are of significantly smaller size than the dataset itself. Also, the model resides in the edge devices and hence internet connectivity is no longer a must during the entire training process.

Compared to the traditional machine learning method where the data from edge devices needed to be sent over to a centralized server for training [5], FL allows edge devices like mobile phones or Arduino to collectively train a shared ML model. Throughout the training process the data never leaves the devices. This approach ensures data privacy and greater enforces security. Using FL, hospitals, banks, and e-commerce companies can collectively train ML models under reduced liability.

FL has been tested on production level on mobile phones and microcontrollers like Arduino and RaspberryPi. Thus, a complex hardware infrastructure is not required to train models anymore enabling access to the data generated by minimal hardware devices [3].

1.2.3 Challenges of Federated Learning

FL comes with its shortcomings too. Communication is one of the core issues in any given FL Architecture. The communication protocol needs to be effective in order to improve the efficiency of the training process. The total number of communication rounds must be kept in check and only the required iterative updates needs to be sent as part of the process.

FL is a new concept and further research needs to be conducted to accommodate for different participating devices. Add to that, it is expected to handle variability in the hardware, including storage, computational as well as communication capabilities of devices because edge devices come in a plethora of shapes and forms. FL systems should also be able to handle a low device participation rate where only a small fraction of the total devices is active at any given moment during a training cycle.

Although FL is designed to protect data from the device, the sharing of model updates can still reveal sensitive information according to research [2].

1.3 Gboard on Android

Google is responsible for one of the leading products of federated learning in the industry. They have applied the concept to its keyboard, Gboard on Android [2].

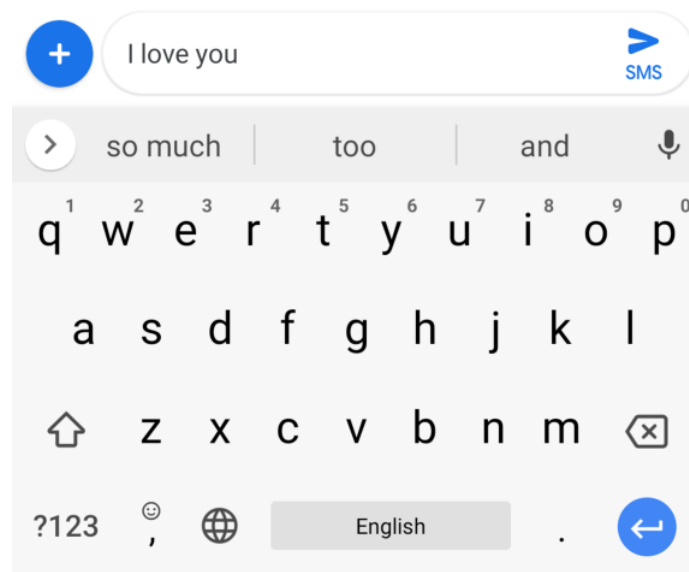


Figure 1.4 Gboard next word prediction in Gboard [4]

When a user enters a word in the Gboard, it suggests a word query as demonstrated in figure 1.5. The device locally stores the data related to the typed words and suggests words to complete the sentence given the context. During the entire process, the typed words are used to train a global model using FL on the device itself. The data never leaves the device. Gboard processes all the history to suggest improvements in the next iteration of Gboard's query suggestion model to its global user [5].

1.4 Outline of the report

This report is structured into seven sections.

First section introduces the concept of FL, three types of FL, benefits, and challenges of FL, and talks about a product leveraging the FL technology, currently being used by Google on Android devices called GBoard.

Second section presents the objectives and motivation behind doing this project.

Third section explains the methodology such as reasons for choosing Horizontal FL as the foundation of the FL platform. The architecture for Horizontal FL is described in detail followed by the security protocols that aids Horizontal FL in maintaining data privacy.

Forth section presents the products developed, research, and results of this project.

Fifth section mentions the difficulties and limitations encountered during the project.

Sixth section briefly mentions the prospects of the final year project.

Seventh section concludes the report. It wraps up the motivation behind this project and the products delivered.

2 Objectives and Motivation

2.1 Objectives

The main objective of this project is to build a federated learning platform where clients can train a machine learning model together on a single platform. The platform should ideally be able to handle Horizontal Federated Learning, i.e., the feature space is same, but the sample space is different.

Subsequently, the platform will be built to accommodate for AI InnoBio's COVID-19 detection machine learning model. However, since the platform is highly dependent on the model itself, we will have to find a different model, if AI InnoBio is not able to provide a model. Once the model provided by the company has reached maturity, we hope to eventually migrate that model onto the platform. After the platform is built, it will be evaluated using three criterion - accuracy, efficiency, and privacy.

1. **Accuracy** – comparing the discrepancy between federated learning and traditional ML.
2. **Efficiency** - speed of training and aggregating the models given an increase in users.
3. **Privacy** - assessing the security of data and the user nodes.

Comparisons and improvements will be researched and made accordingly to aforementioned criteria.

2.2 Motivation

AI InnoBio Limited is a Biotechnology Start-up in Hong Kong that utilizes an CMOS sensor technology to develop a novel hand-held spectrometer device. They hope to use our federated learning platform with AI models to conduct saliva tests to determine in less than a second, whether a certain patient is infected with the coronavirus. A hospital in Israel conducted clinical trials with hundreds of patients with this new artificial intelligence-based device and is able to achieve a 95% success rate of identifying coronavirus in the body [\[6\]](#).

AI InnoBio hopes to expand their business in Asia to allow different countries to use the device to perform a fast, accurate and low cost COVID-19 detection test. To obtain a machine learning model with better accuracy, we need to obtain more test data across different countries. Data across several demographics will help generalize the ML model and ideally yield more accurate predictions. The process of collecting data is where the problem of data privacy lies. Different countries and hospitals are not willing to share their test results and related patient information, and therefore it is difficult to utilize the traditional machine learning method, where the entire dataset needs to be centralized, to train the model. Hence, a platform is required to perform federated learning with different clients to improve the accuracy of the model while keeping the data privacy intact.

This project aims to lay the groundworks for a FL platform which will aid the rapid testing of COVID-19 in countries that have huge numbers of suspected cases like India. Besides making the platform, most of our time and effort in this project will go towards the research in the field of federated learning. We will conduct research of different frameworks, algorithms, and methodology. Moreover, it is also hoped that this project could also provide a foundation for future projects with the same nature of data privacy.

3 Methodology

3.1 Introduction

Federated Learning can be categorised into three types with contrasting system architectures as they deal with different sample space and feature space. This section aims to explain why Horizontal FL was picked for the project and provides a detailed presentation of the architecture of a typical Horizontal Federated Learning platform.

3.2 Rationale for using Horizontal Federated Learning

Horizontal Federated Learning is the most common type of federated learning in the market compared to its two other counterparts, vertical and transfer learning [2]. It is easier to find open-source works related to it which reduces the time of development of the platform.

As mentioned before, the platform is expected to accommodate the COVID-19 detection ML model of AI InnoBio. However, during the time of our project, the model was not developed yet and hence, speculations needed to be made for the model and datasets to be used. The CMOS sensor device delivers an array of length 1024 for each sample. Since the method of detection is the same, the feature space is the same. But the data for training the model will come from different geographical regions, making the sample space different. Since the requirements fulfil that of Horizontal Federated Learning, hence we used its architecture to form the foundation of our project.

3.3 Horizontal Federated Learning Architecture

In this section, the architecture of the Horizontal Federated Learning Platform will be discussed as it forms the base of our FYP (shown in Figure 2.1). It is similar to the one in section 1.2.1 but this subsection will delve deeper into the technical details of each step involved in the process.

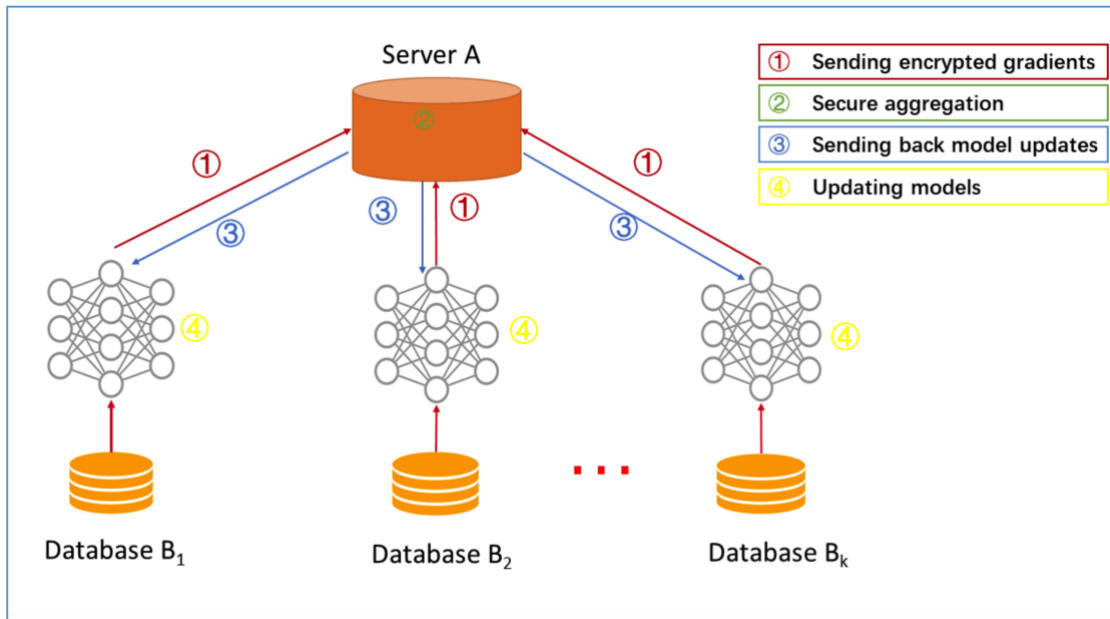


Figure 3.1 Architecture of Horizontal Federated Learning System [2]

The architecture design shown in Figure 2.1 will be followed in order to build the federated learning platform. In this architecture, k participants (clients) collaboratively train the model located in the central server A. It is assumed that the participants are all honest, i.e., the participants are not eager to learn extra information besides what they already know, and that the host server may be susceptible to snooping internally. Hence, information leakage should not take place during communication between the clients and the server, but we will prepare for internal data compromises.

The training process of a typical system is as follows [2]:

- **Step 1:** training gradients are locally computed in the participants; a selection of gradients is masked using encryption, differential privacy, or secret sharing; the masked gradients are sent to the server.
- **Step 2:** aggregation is performed in the server securely without leaking any information about the participants.
- **Step 3:** aggregated result is sent back by the server to the participants.

- **Step 4:** the gradients are then decrypted and are used by the participants to update their models, respectively.

Until the loss function converges, the above steps are continuously iterated. This architecture is independent of the machine learning algorithm and it will be followed when building the platform for COVID-19 detection as the architecture only focuses on the sharing of updated model parameters during the training stage.

3.4 Security Protocol

Privacy of data is the focus of Federated Learning and is the reason why the concept became popular in the industry. Security models and protocols are the heart of FL that provide privacy guarantees. In Horizontal Federated Learning process, it is assumed that the participants training the model are honest and security measures need only be implemented on the server side which might be honest but curious [2]. This means that, only the server can compromise the data owned by the participants.

This subsection briefly explains three popular security protocols - Secure Multi-party Computation, Differential Privacy, and Homomorphic Encryption.

3.4.1 Secure Multi-party Computation

In Secure Multi-party Computation, the goal is to create methods for different participants to jointly compute a given function. The inputs are provided by the respective participants and hence are kept private from each other. Zero information leakage is desirable but requires complicated computation intensive protocols. Therefore, for practical purposes partial information disclosure is generally accepted given security guarantees of the disclosed information [2].

3.4.2 Differential Privacy

In Differential Privacy, generalization methods are used to obscure sensitive fields. Noise is added to the data so that it is hard for third parties to distinguish the data that participants are referring to. This makes the data impossible to recover just by seeing the noisy output and hence protects user's privacy. Differential Privacy has its shortcomings as it can possibly affect the accuracy of the ML model [2].

3.4.3 Homomorphic Encryption

Homomorphic encryption involves parameter exchange during machine learning. This form of encryption allows one to perform operations on the encrypted data directly without having to decrypt it first. The decrypted result of the operations should theoretically be same as the result of the operations had it been performed on unencrypted data. Neither the model nor the data is transmitted and hence it is impossible for other parties to guess the data. This results in zero information leakage [\[2\]](#).

3.5 Summary

The security protocols were not the priority in our project. But they were researched on which helped us compare and select the framework suitable for our platform. Since, FL is new, many frameworks do not focus on the implementation of these protocols yet, rather they focus on the effective communication between the server and the clients. In the future, it will be possible to leverage the three protocols to enhance the security and data privacy.

This section explained the rationale behind picking the Horizontal FL architecture for the project. It also described the security protocols that are useful in maintaining data privacy of edge devices. The next section will report the project's experiments and subsequent results of the project.

4 Experiments and Results

4.1 Overview

The following chapter presents our work and subsequent results obtained during the completion of this project. Extensive research was conducted on four different open-source frameworks, namely, Tensorflow Federated, Flower, PySyft and Sherpa AI. PySyft was used alongside couple other frameworks – PyGrid and Syft.js, to leverage the higher-level APIs.

4.2 Tensorflow Federated

TensorFlow Federated (TFF) is a machine learning framework that can perform computations on decentralized data. An algorithm can be trained using TFF across edge devices holding local data samples, without ever exchanging the data samples. It was developed by Google to facilitate the research on FL. TFF allows developers to simulate FL on the ML models.

While experimenting with TFF, each dataset was used for training using 3 different methods. Firstly, the traditional machine learning algorithm was implemented which runs on centralized data. The results of this method acted as the baseline for comparison for the other 2 methods. Secondly, TFF Learning API was leveraged. It offers a set of high-level interfaces that could be applied for the implementations of federated learning. The model created using Keras was wrapped by `tff.learning.Model`, which was then accessed using `tff.learning.API`. Lastly, TFF Core API was used for the third method. This allowed us to express the novel federated algorithms by combining Tensorflow with communication operators in a distributed setting. In this method, 4 functions were defined which are as follows:

1. Server-to-Client Broadcast – broadcast the weights of the shared model to the clients.
2. Client Update – compute the updated weights in the clients.

3. Client-to-Server Upload – send the updated weights of the clients to the server.
4. Server Update – average the weights and update the shared model in the server.

Three datasets namely Fashion MNIST, CIFAR10 and Cats_vs_Dogs were picked to perform the research on this framework. The results listed in the following sections.

4.2.1 Fashion MNIST

Fashion MNIST is a dataset of Zalando’s article images. It comprises of 60,000 training examples and 10,000 testing examples. Each of the example is a grayscale image of size 28x28 pixels, associated with a label from 10 classes. Figure 4.1 shows example images present in the dataset.

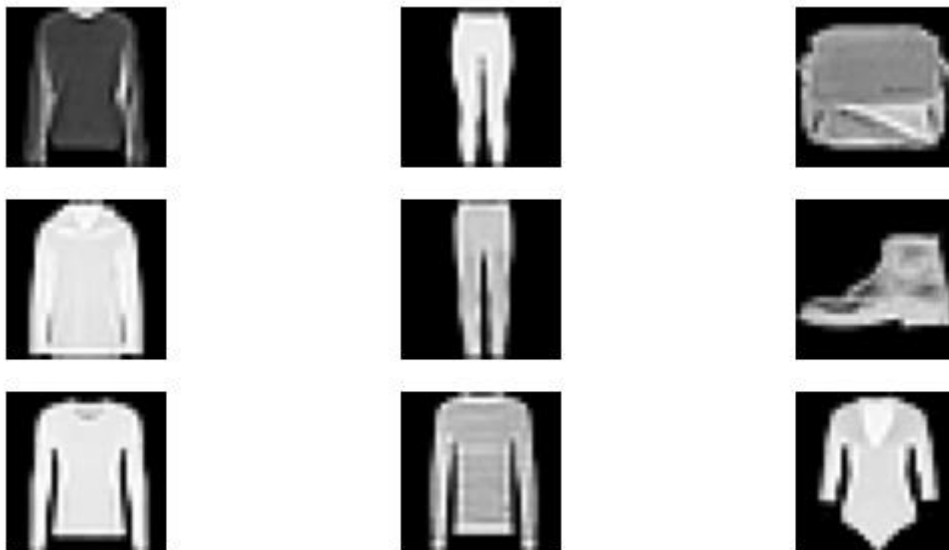


Figure 4.1 Sample images from Fashion MNIST

Preprocessing of the images was done before being fed into a fully connected neural network (FCNN). The figure below shows that the images were flattened into an array of length 784 (28x28) which was then directed into a dense layer comprising of 128 neurons. The final layer had 10 neurons which pointed to the class of the image fed into the FCNN.

```

Model: "sequential"

```

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 128)	100480
dense_1 (Dense)	(None, 10)	1290

```

Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0

```

Figure 4.2 Neural network used for Fashion MNIST

Firstly, the model was trained using the traditional method of machines learning where the image was stored in a single machine. 10,000 data samples were used in the training process which lasted for 10 epochs. The validation and training loss is shown on the left while the validation and training accuracy is shown on the right in Figure 4.3. The validation accuracy was 84%, acting as the baseline for the remaining two methods.

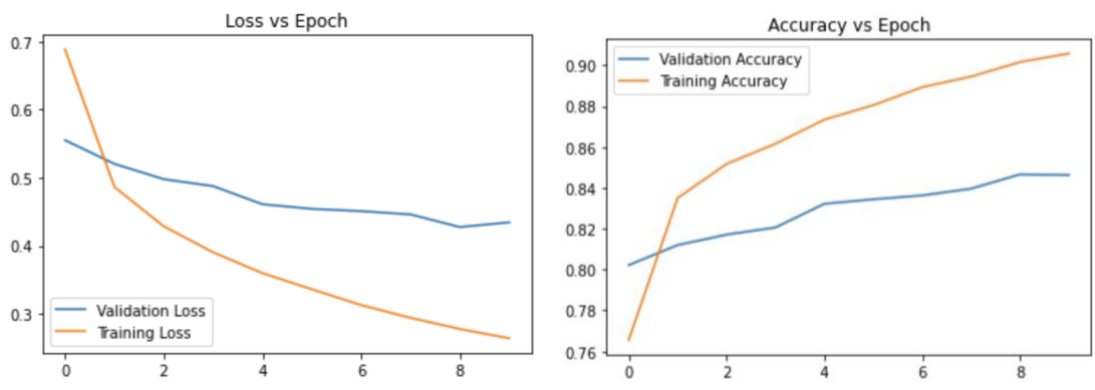


Figure 4.3 Loss and accuracy of neural network trained with traditional machine learning technique on Fashion MNIST

Secondly, the model was trained using the higher-level APIs provided by TFF Learning API. The decentralized environment was simulated on the same machine by separating the 1,000 images into 100 clients randomly. Figure 4.4 displays the class distribution of 3 clients that participated in the training process. As seen in the figure, the distribution of images based on classes is indeed random.

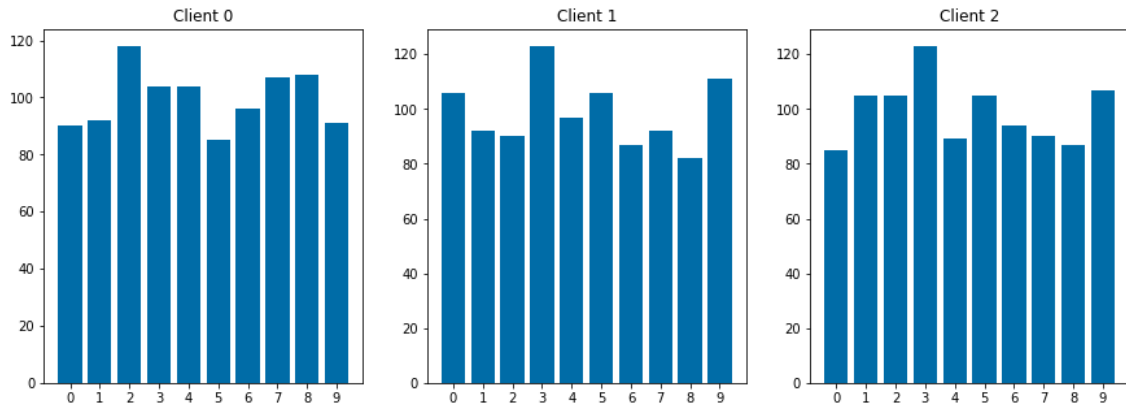


Figure 4.4 Class distribution of Fashion MNIST

In the training stage, all the 10 clients participated in every round. In total, 10 rounds of training were conducted.

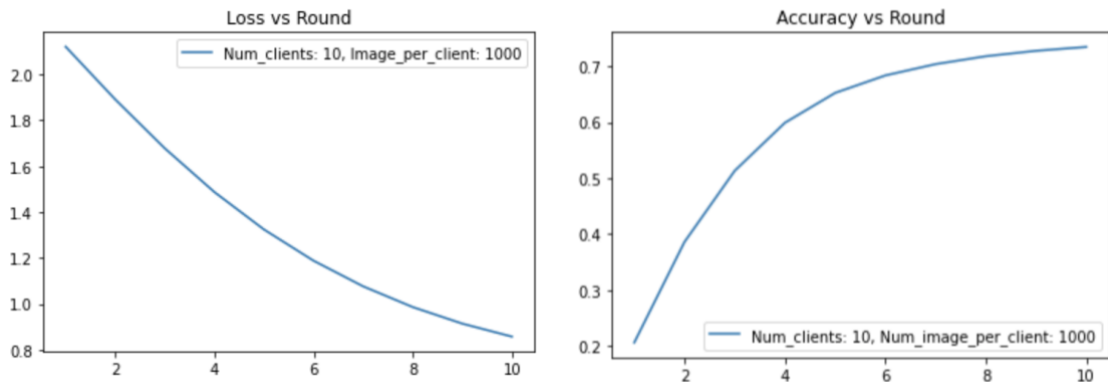


Figure 4.5 Validation loss and accuracy of neural network trained with Tensorflow Federated Learning API on Fashion MNIST

Figure 4.5 shows the gradual drop in the loss and the subsequent increase in the accuracy of the model over the 10 rounds of training. This proves that the FL algorithm was indeed capable of training the model. The validation accuracy reached 74%. This is lower than the 84% achieved in the first method. It is to be noted that ideally the accuracy of both the algorithms are supposed to converge if the training is done for enough rounds/epochs.

Lastly, using TFF Core module, the 4 steps were custom built.

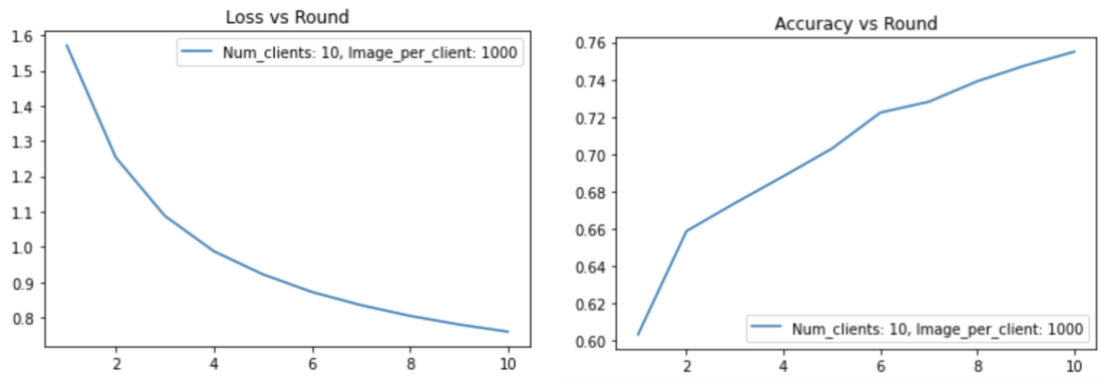


Figure 4.6 Validation loss and accuracy of neural network trained with Tensorflow Federated Core on Fashion MNIST

The training was run for 10 rounds with 10 clients with 1,000 images each in each round. The accuracy of 75% was obtained as shown in Figure 4.6 comparable to the second algorithm.

4.2.2 CIFAR10

CIFAR10 was developed by Canadian Institute for Advanced Research. It is a collection of 60,000 32x32 pixel images. The 60,000 images are split into 50,000 training images and 10,000 testing images. Similar to the Fashion MNIST dataset, it has 10 different classes representing airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The figure below displays 9 images present in the dataset.

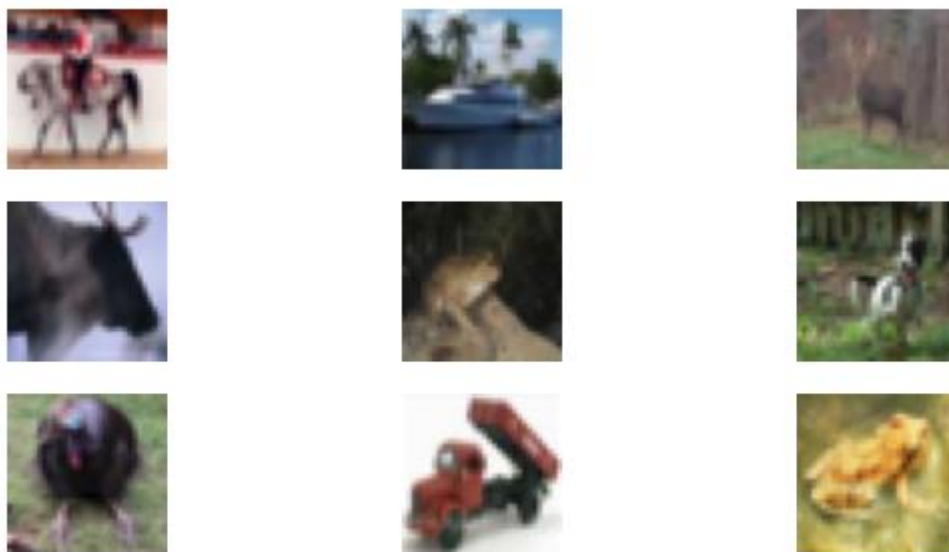


Figure 4.7 Sample images from CIFAR10

To achieve a desirable accuracy, the model was changed from pure FCNN to a combination of convolutional neural network (CNN) and FCNN. CNN was used to capture more spatial information present in the image. A max_pooling layer was added after each conv2d layer as shown in Figure 4.8.

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36928
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65600
dense_1 (Dense)	(None, 10)	650

Total params: 122,570
 Trainable params: 122,570
 Non-trainable params: 0

Figure 4.8 Neural network used for CIFAR10

The Traditional method of training yielded a validation accuracy of 55%. This is lower than the accuracy achieved using the Fashion MNIST dataset. It is speculated that since CIFAR10 consists of coloured images it proves to be a harder problem to classify than the Fashion MNIST.

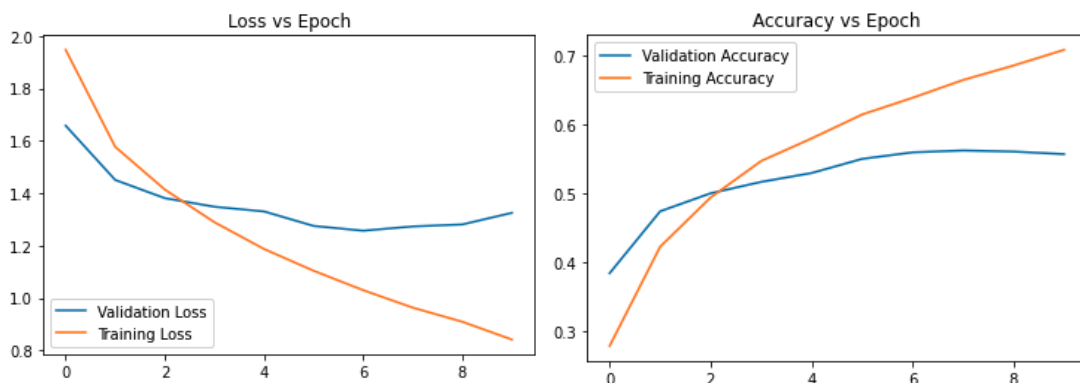


Figure 4.9 Loss and accuracy of neural network trained with traditional machine learning technique on CIFAR10

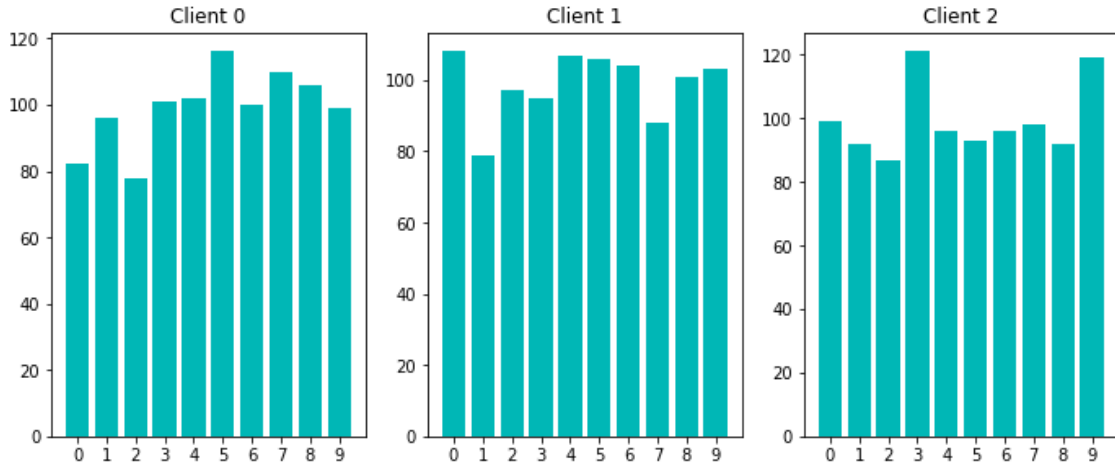


Figure 4.10 Class distribution of CIFAR10

Next, to train the model simulating federated 1,000 images were distributed among 100 clients randomly. Figure 4.10 displays the class distribution of 3 clients that participated in the training process.

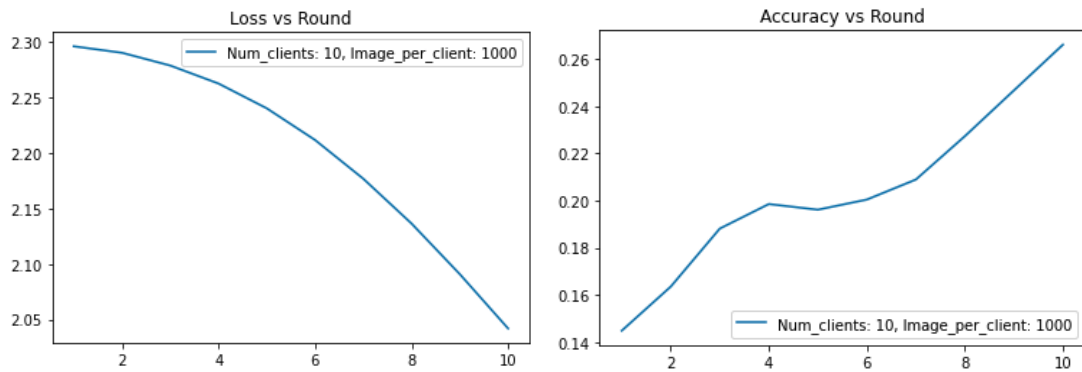


Figure 4.11 Validation loss and accuracy of neural network trained with TensorFlow Federated Learning API on CIFAR10

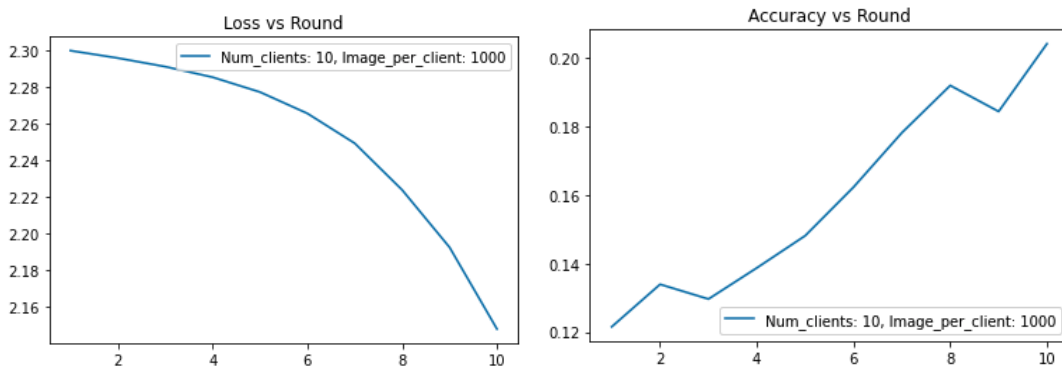


Figure 4.12 Validation loss and accuracy of neural network trained with Tensorflow Federated Core on CIFAR10

The validation accuracy of 26% is lower than the 55% achieved in the traditional method. CIFAR10 requires significantly more number of rounds to converge to the accuracy attained by the centralized algorithm. It is also interesting to notice that the validation accuracy was not increasing between rounds 4-6 (Figure 4.11).

4.2.3 Cats_vs_dogs

Cats_vs_dogs was picked for was specific reason. As mentioned before in our motivation, the project was undertaken to help AI InnoBio to help develop a platform for COVID-19 detection. Cats_vs_dogs is a binary classification problem similar to COVID-19 detection, where the output is either positive or negative. Experimenting with this dataset proves that TFF is capable of handling binary classification in federated setting. Cats_vs_dogs is a dataset consisting of 25000 images of various sizes. Each image is coloured having 3 channels.



Figure 4.13 Sample images from Cats_vs_dogs

Before the data was fed into the model for training, it was pre-processed to ensure uniformity. The images were also altered a bit to induce noise in the dataset.

Traditional Machine Learning algorithm was first implemented to set the baseline for this dataset.

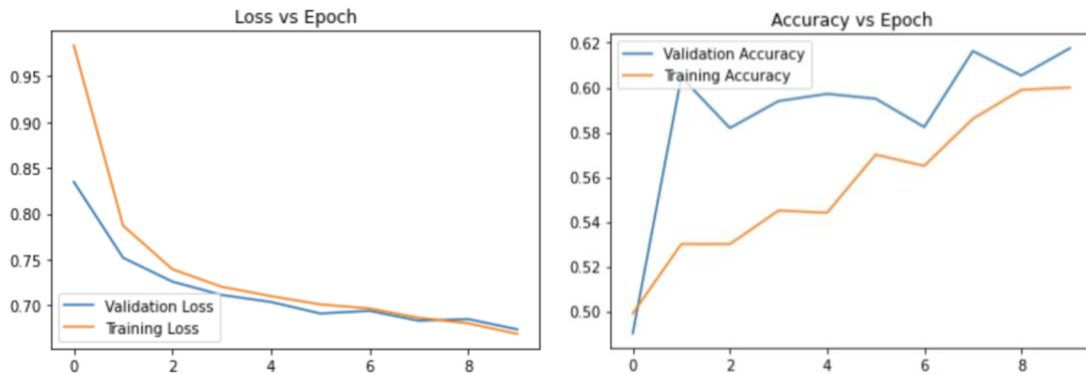


Figure 4.14 Loss and accuracy of neural network trained with traditional machine learning technique on Cats_vs_dogs

Cats_vs_dogs is a classification problem of similar difficulty as CIFAR10, but it has only two classes. Hence, the validation accuracy of 62% (lower than Fashion MNIST, but higher than CIFAR10) is expected.

TFF Learning API was utilized for on-device simulations. Again, 100 clients were initialized with 1,000 images. The distribution based on the class of 3 clients are shown in the figure below.

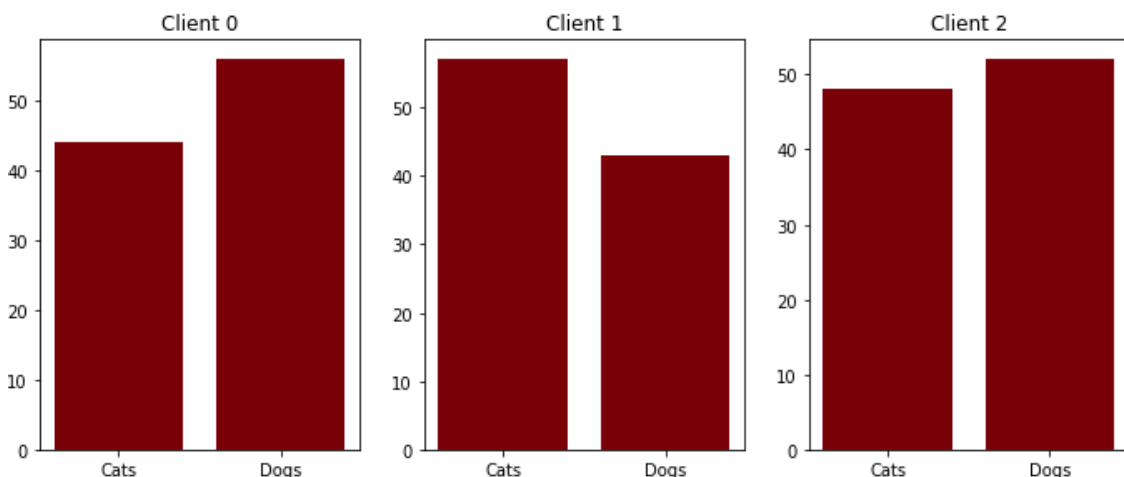


Figure 4.15 Class distribution of Cats_vs_dogs

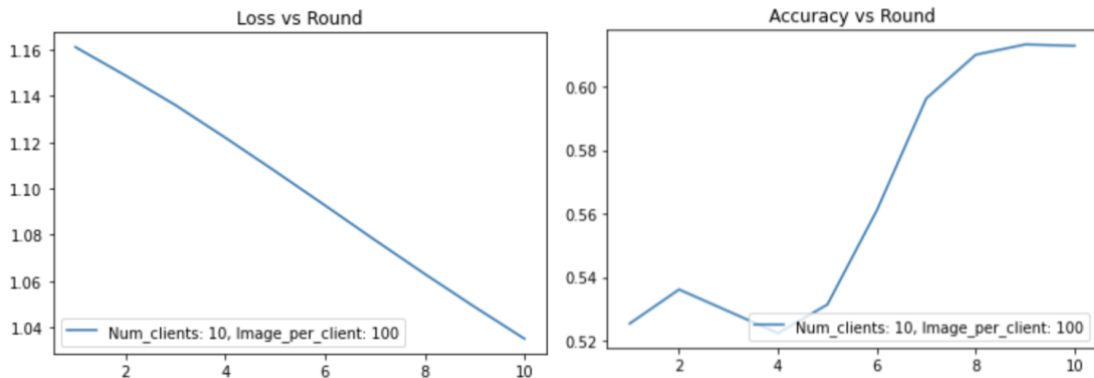


Figure 4.16 Validation loss and accuracy of neural network trained with Tensorflow Federated Learning API on Cats_vs_dogs

Figure 4.16 shows the validation accuracy obtained over the 10 rounds of training the model. The accuracy converged at 62%, same as the traditional machine learning model. This shows that 10 rounds is enough for the federated learning to reach the ideal accuracy, which is an exciting result.

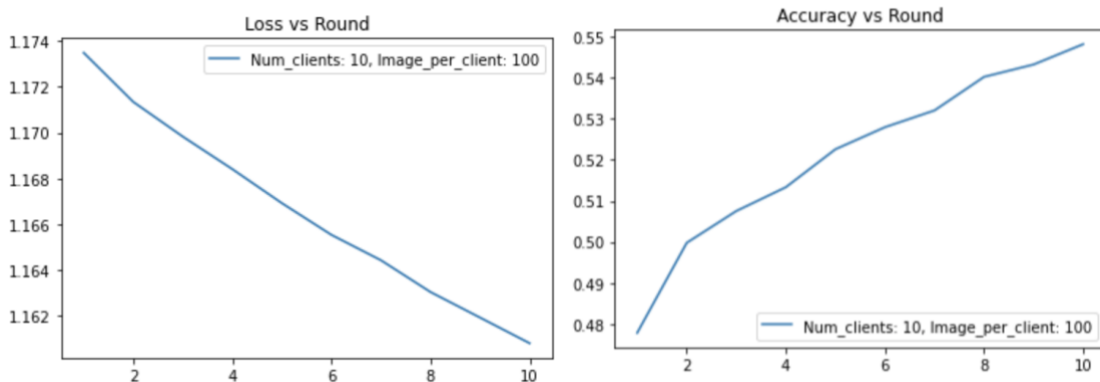


Figure 4.17 Validation loss and accuracy of neural network trained with Tensorflow Federated Core on Cats_vs_dogs

Finally, similar to Fashion MNIST and CIFAR10, the dataset was used for training by self-implementation of the 4 steps leveraging TFF Core. The accuracy of 55% is lower than the previous two algorithms, speculating that 10 rounds is not enough for the convergence of the accuracy. Also, the difference in complexity of the datasets can be a possible explanation of the difference in accuracy.

4.2.4 Hyperparameters Comparison

To gain a better understanding about the relation between the hyperparameters, experimentation was conducted in a decentralized setting. For the experiments, the Fashion MNIST dataset was used.

The first experimentation was done to explore whether choosing different clients would affect the accuracy obtained in federated learning. Hence, in the training one model was trained using the same 10 clients while another was trained using 10 randomly selected clients for every round. It can be observed from Figure 4.18 that the validation loss converges quicker when different clients are selected. It can be speculated that different clients give the model a chance to look at more data and hence have a better capability of generalization. The validation accuracy is higher explaining the speculation of generalization.

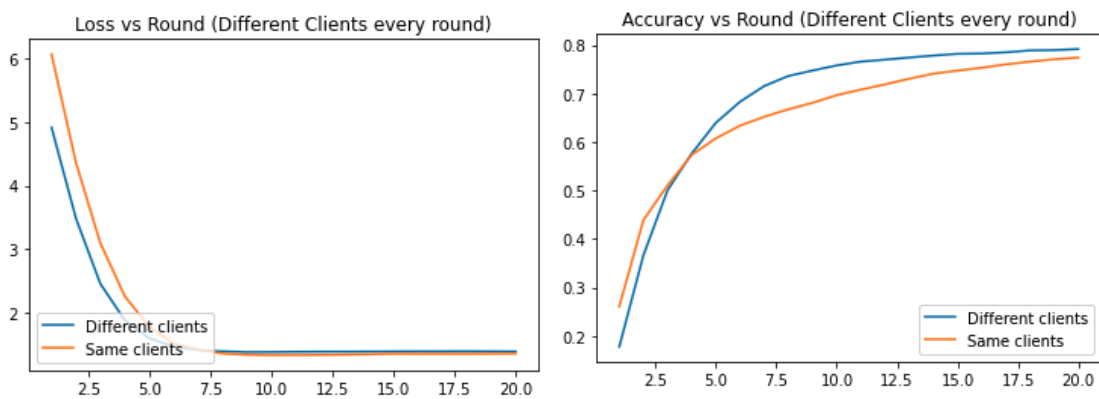


Figure 4.18 Validation loss and accuracy of neural network trained with the same/different clients for each round

Next we experiment by changing the number of client and images per client to see its effect on the validation accuracy and loss. We use the grid search method to find the optimal hyperparameter values. It is a general trend that increasing the number of clients for constant images per client results in higher accuracy and lower loss.

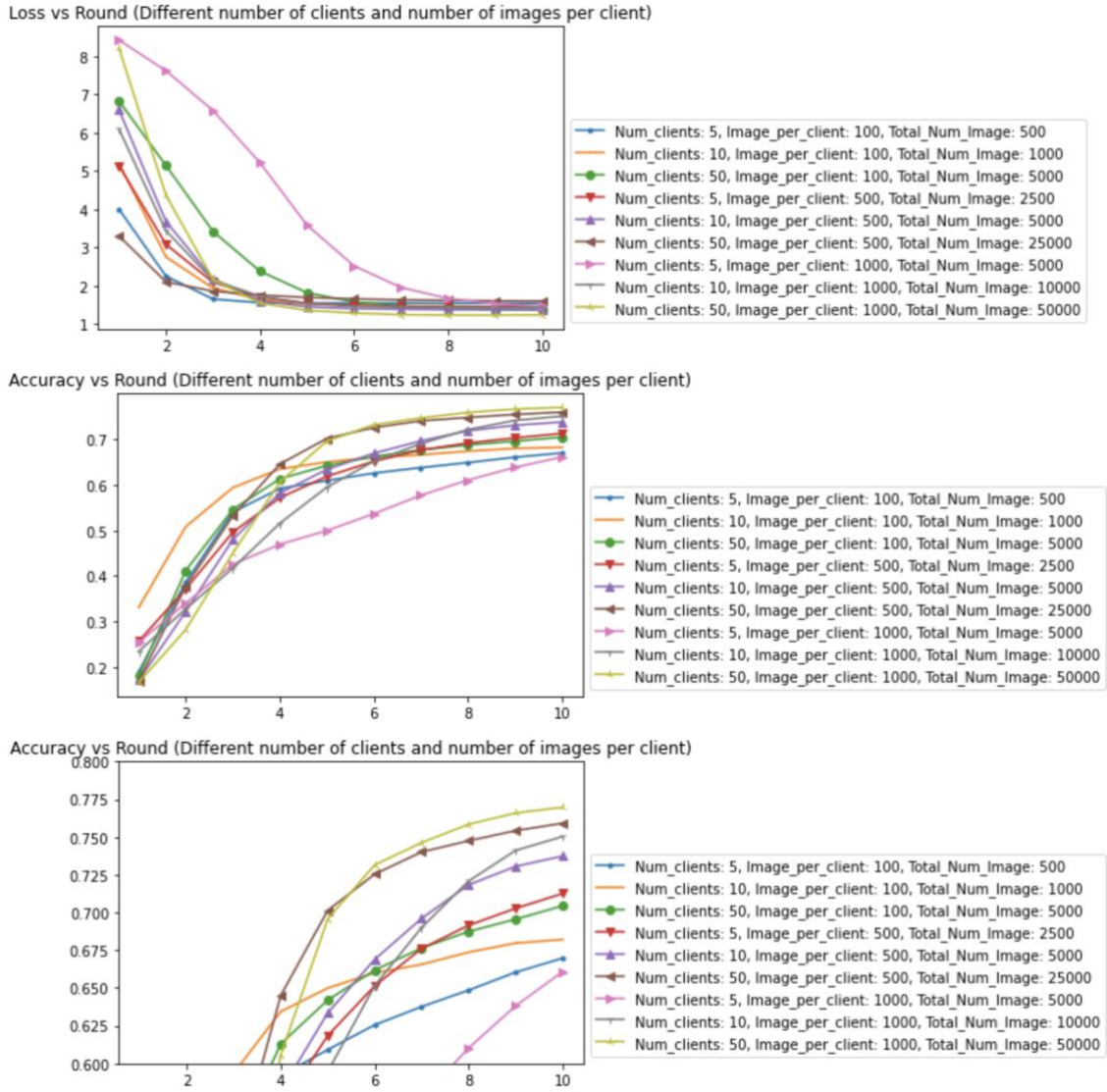


Figure 4.19 Validation loss and accuracy of neural network trained with different number of clients and number of images per clients

Finally, the total images constant (5000) was kept constant. In this experiment, the images were distributed using the following formula:

$$\text{Total number of images} = \text{Number of clients} * \text{Images per client}$$

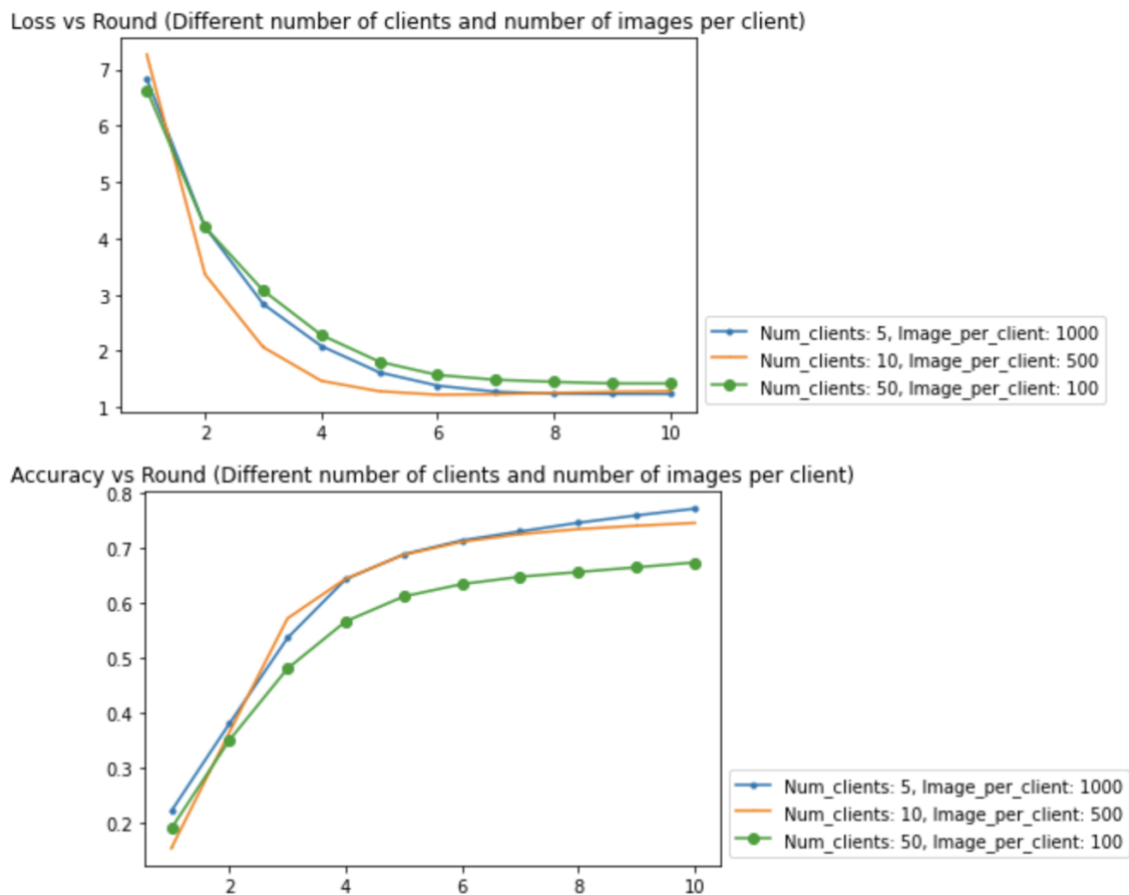


Figure 4.20 Validation loss and accuracy of neural network trained with 5000 images in total

From Figure 4.20, it can be observed that less clients with more images yield better accuracy than having less images distributed across various clients. This is an expected outcome as decreasing the number of clients converges towards traditional machine learning method where there is only 1 client with all the images present in it. And, from previous experimentation with Fashion MNIST, CIFAR10 and Cats_vs_dogs, traditional machine learning always had the highest accuracy.

4.2.4 Limitations

TFF is a comparatively new framework. It is mostly meant for experimentation purposes and hence does not support the capability of production level deployment. Needless to say, federated simulations can run on a single runtime container on Google Cloud Platform and docker. Upon future development of TFF by Google, it might be possible to deploy the code on servers.

4.3 Flower

The next open-source framework that was investigated during this project was the Flower framework. Flower was developed in to bridge the gap between FL research and production. A federated learning platform was developed in Python programming language over the last semester.

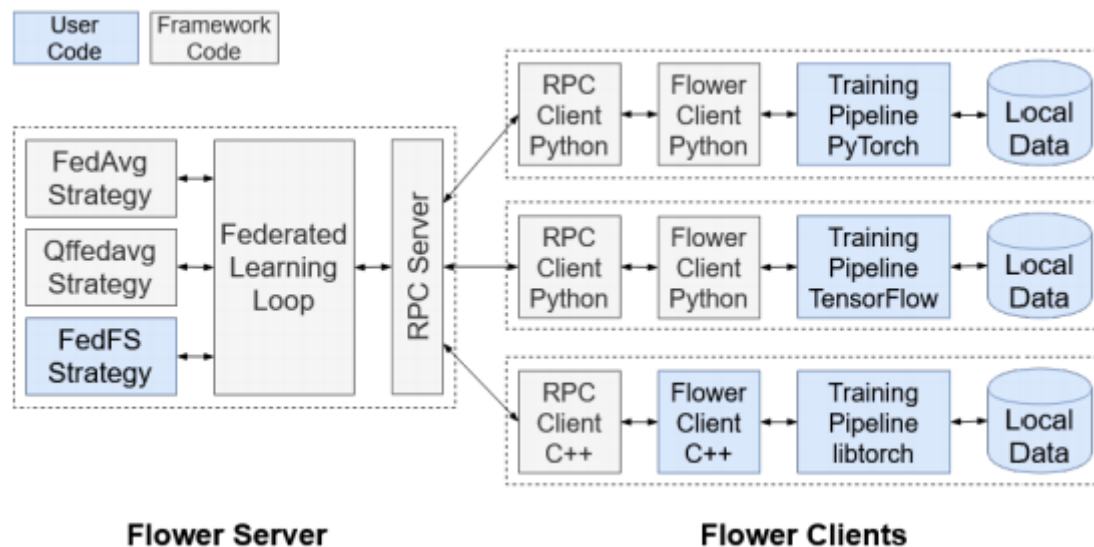


Figure 4.21 Architecture of the Flower Framework

The platform comprises of a server and multiple clients. They interact with each other during the training process over a gRPC connection. The server accepts the clients into the training process and is responsible for overseeing the participation of clients. It selects specific clients each round depending on the requirement. The client is responsible for connecting to the server and training the shared model using the local data. The client sends over the updated model weights to the server for aggregation. The framework handles the underlying communication between the server and the client and gives developers the ability to design custom training algorithms.

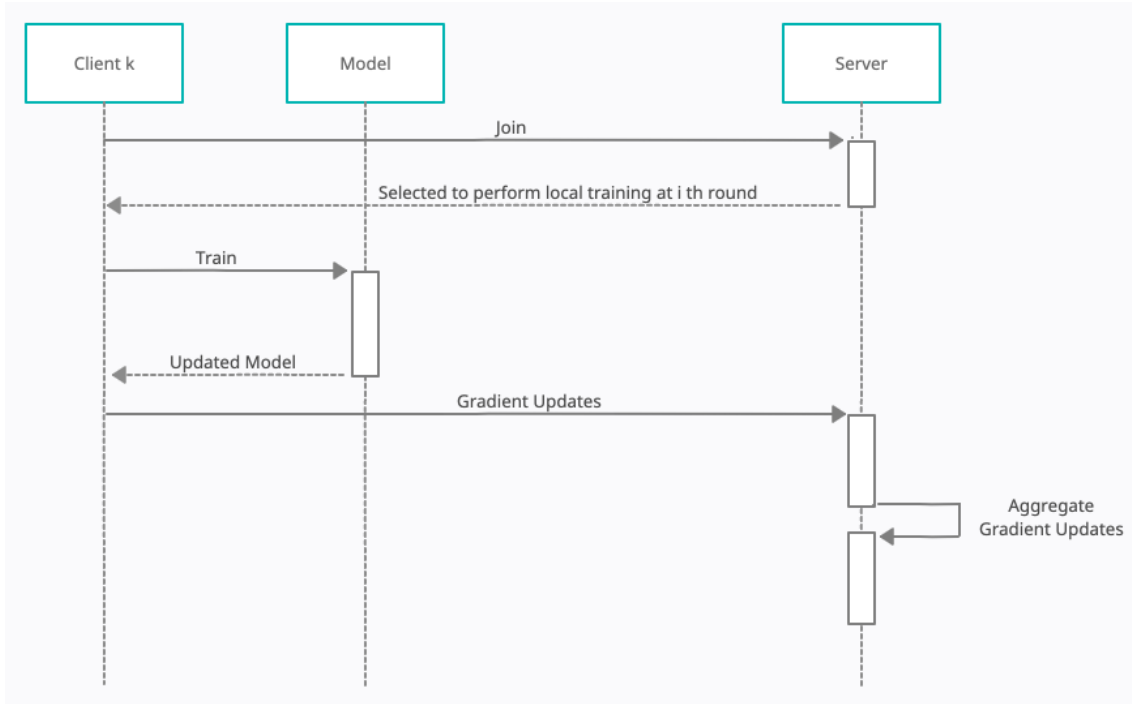


Figure 4.22 Sequence Diagram of one round in the FL CLI platform

Figure 4.22 shows the flow of execution for a typical round of training using the FL framework.

4.3.1 Use Cases

The 4 use cases of the Flower framework are listed as follows:

1. Reproducible research – The framework allows researchers to focus on the problem at hand and not worry about the implantation of the architecture. It enables researchers to experiment and implement new ideas on top of the stack already provided by the framework [7].
2. Federating existing workloads – Flower has the potential to link existing ML frameworks widely available in the market and provide opportunities to run both on mobile and cloud. Using Flowers, developers can translate the existing code in traditional ML to its federated counterpart [7].
3. Heterogenous workloads – Most of the frameworks have limited support for heterogeneity and focus on a particular platform instead. Flower is

robust and supports vastly different client environments. Flower assumes heterogeneity in devices and deals with it behind the scenes [7].

4. Scaling research – Flower aims to enable research which leverages large number of clients training concurrently. This enables researchers to simulate the properties of the real-world FL better [7].

4.3.2 Rationale behind using Flower

The framework was chosen because it is highly scalable, is compatible with edge devices, has a proven infrastructure, and is easy to use and extend [7].

Scalability – Flower is a proven framework that can handle huge number of clients, as much as 10,000. It simulates real-world setups to better fit the developer’s need.

Compatibility – Flower can run on unconventional edge devices like Android, iOS, Raspberry Pi and Nvidia Jetson. In addition, it is accessible on a plethora of operating systems and hardware platforms to work with heterogeneous edge device environments.

Infrastructure - Flower ensures low engineering effort for the creation of the platform which allows the developers to focus on the ML problem rather than the nuances of federated learning.

Usability – Flower is easy to comprehend for beginners as it allows them to write custom algorithms that fulfils the purpose of the desired federated learning platform.

4.3.3 CLI Platform and Evaluation

Leveraging the framework, a CLI platform was built. The MNIST and CIFAR10 datasets were used to test out the platform locally. The platform comprises of a SERVER and a CLIENT folder. In the server folder, there are 3 files. The server.py is the heart of the FL platform. It is responsible for starting the server, broadcasting model weights to the clients, selecting the clients for training and finally aggregating the weights after each round. The aggregation strategy used was FedAvg, which is the most basic form of averaging where the simple mean of updated weights is put into the shared model. The server.py has a customized

evaluation function which tests the updated model after each round of training and saves the model locally so that it can be later used for inference. The predict.py file is used for the inference stage, after the model has been fully trained. The client.py file present in the CLIENT folder is responsible for starting a new client and connecting to the server. It trains the model locally as and when it gets selected for the training process. The client also handles sending model updates to the server file. Both the client and the server have a file where the model is defined and, certain helper functions which ease the process of accessing the training and testing data. In case of MNIST, the file is called MNIST.py, while in case of CIFAR10, the file is called CIFAR10.py.

The hyperparameters of server.py and client.py can be changed according to the requirement of the company. By default, the server selects at least 10% of the available clients, with a minimum of 2 clients per round. Add to that, the FL platform needs at least 3 clients to be connected to the server for FL to begin.

Table 4.1 shows the difference in accuracy and the total training time for the two datasets. In this experiment, 5 clients were used with a minimum of 3 getting selected every round. Each of the clients were initialized with 500 training samples and 10 testing samples using the MNIST.py and CIFAR10.py file, respectively. The test was carries out in the Command Line Interface on a local machine.

Dataset	Model	# rounds	Accuracy	Time (Efficiency)
MNIST	Fully Connected Neural Network (FCNN)	10	85%	24 sec
CIFAR10	Convolutional Neural Network (CNN)	10	29%	1 min 46 sec

Table 4.1 Evaluation Results using the CLI platform (1)

The MNIST dataset and CIFAR10 both comprise of 10 classes and hence it is comparable. MNIST contains grayscale images, unlike CIFAR10 which contains RGB images, with 3 channels. Since, all the other factors were kept, we can argue that training done on CNN is slower as compared to FCNN as CNN requires considerably more computations in every round. FCNN has a smaller number of

trainable weights than CNN and hence takes less rounds to converge. Since, the rounds were kept constant, MNIST model reached 85% accuracy which is clearly higher than the 29% of CIFAR10.

All the following experimentation was done on the MINIST dataset.

# Clients	# Rounds	Accuracy
5	5	82.11%
5	10	85.50%
5	15	86.04%
5	20	86.47%

Table 4.2 Evaluation Results using the CLI platform (2)

In the first experiment, the number of clients was kept constant. Only the number of rounds was altered to see its effect on the accuracy. From Table 4.2, we notice a considerable jump in the accuracy from 5 rounds to 10 rounds. But after that, the accuracy seems to converge and if we keep on increasing the number of rounds, we can extrapolate the accuracy to be around 87%.

# Rounds	# Clients	Accuracy
10	2	81.67%
10	3	84.15%
10	4	84.94%
10	5	85.50%

Table 4.3 Evaluation Results using the CLI platform (3)

Next, the minimum fit clients per round was kept constant at 2 (for better comparison) along with the number of rounds at 10. We notice the same trend as the previous experiment, where the accuracy jumps from 2 to 3 clients and then seems to converge. It depends on the future users of the platform to choose the optimal number of rounds and clients according to their business requirements.

Unnecessarily increasing those two hyperparameters might not be worth the training cost in terms of time and computation.

# Rounds	# Clients	# Minimum Fit Clients	Accuracy
10	2	2	81.67%
10	3	3	81.83%
10	4	4	84.25%
10	5	5	83.81%

Table 4.4 Evaluation Results using the CLI platform (4)

In the final experiment, only the rounds were kept constant, and the number of clients and number of minimum fit clients were chosen to be the same. Table 4.4 suggests that for 10 rounds of training 4 clients, all participating in all the training rounds is the optimal choice, as increasing the number of clients results in a decrease in the accuracy. The increase in the data seen by the model due to increase in the clients is countered by the noise created by training the models separately on the different clients. This result shows that the future user of our platform needs to find the right balance to get the most out of the training process.

4.3.4 Deployment

The platform currently only runs on the CLI on the local machine and has not been deployed yet. The flower framework is currently in its early stage and is only capable of on-device simulations. Also, extensive testing needs to be done to check for data leakage during the communication between the clients and the server before it gets deployed. Only then will our product pass the standard of production.

4.3.5 Difficulties

To overcome the deployment limitation of Flower framework, the CLI platform was tested using a pipeline tool called Ngrok. Ngrok allows the port on which the server is from to be exposed to the internet. Hence, the rationale behind using

Ngrok was to check whether the server running in the local machine can be connected to the clients on another machine. Unfortunately, the test failed as the client was not able to connect to the server even with when the ports were exposed to the internet. The reason is still unknown, but it is speculated that it related to the communication stack of the framework which used gRPC protocol. Further research needs to be conducted to get to the crux of this problem and only then can we deploy our platform on multiple machines.

4.4 PySyft, PyGrid and Syft.js

The third framework that was experimented with, was a combination of three frameworks – PySyft, PyGrid and Syft.js. The following section will delve deeper into each of the three frameworks.

4.4.1 PySyft

The PySyft library was developed to ensure secure and private ML. It decouples the private data from the training process of the models, using FL, Differential Privacy, Multi-Party Computation and Homomorphic Encryption. It is built on top of PyTorch and Tensorflow and hence the code written for traditional ML can be easily translated.

In the Syft ecosystem, one can write software which is capable of computing information that is not owned by the machine and hence have no control over it. This allows the data to reside in its ownership while allowing it to be used for computation in a private manner.

PySyft can perform two major computations:

1. Dynamic - directly compute private data securely.
2. Static - create static computational graphs that can be deployed and scaled as desired on different compute.

4.4.1.1 Duet

Duet is a software developed on top of the PySyft framework. It is a peer-to-peer tool that allows the Data Owner (hospital in our case) to expose the data to the Data Scientist (developer at AI InnoBio). All this happens in a private setting. The Data Scientist can manipulate the data through a zero-knowledge access control mechanism. This mechanism can be used in our case to design a 1-to-1 ML model training platform. Unlike usual FL, where there are several clients training the model. The product made using Duet allows for each data owner to train the platform one by one. Also, the Data Owners have the right to decide whether to participate in the training process and allow any manipulation of data. The product developed using duet can not only be used for ML, but also allow data scientist to gain invaluable insights from the data owned by the Data Owner. The most impressive part about Duet is that the data never leaves the Data Owner and hence, data privacy will never be an issue.

The Duet platform was used to demonstrate the training of a ML model using PySyft without being deployed in PyGrid ecosystem. Thus, the data and model need not be hosted in the first place. As Duet only allows one client server to communicate with each other over a secure connection, we cannot create multiple clients and train concurrently. Hence, while experimenting with Duet, we simulated multiple clients. We trained the model using the MNIST dataset. In the first experiment, we made the single client hold all the 60,000 training images, while in the second experiment, we simulated multiple clients by training the model in multiple stages. For each stage, the model is trained with the part of data that is present in each client.

# Clients	# Data per client	# epochs	Accuracy	Time (sec)
1	60000	5	97.86%	1077
5	12000	5	96.71%	1268
10	6000	5	95.29%	1564

Table 4.5 Results of simulating multiple clients (5 epochs)

# Clients	# Data per client	# epochs	Accuracy	Time (sec)
1	60000	10	98.00%	2156

5	12000	10	96.46%	2588
10	6000	10	94.48%	3139

Table 4.6 Results of simulating multiple clients (10 epochs)

Table 4.5 shows the accuracy and training time using the Duet platform when the training lasted for 5 epochs. Table 4.6, on the other hand, displays the results when the experiment was conducted for 10 epochs.

4.4.2 PyGrid

PyGrid is a python library that provides APIs for the management and deployment of PySyft at scale. It is a peer-to-peer network of data owners and scientists that can collectively train ML models. It enables the extension of PySyft to perform FL on jupyter notebook, web, mobile, and other edge devices with different Syft worker libraries.

PyGrid is composed of three components:

1. Network – manage, monitor, control, and route instructions to various PyGrid domains.
2. Domain – store private data and models for federated learning, as well as to issue instructions to various PyGrid workers.
3. Worker – ephemeral instance that compute data and is managed by the domain.

In this project, we are dealing with the architecture of Horizontal Federated Learning, and hence only PyGrid domain is required.

4.4.2.1 Creation of Plans using PySyft

Besides, being used for developing the Duet platform, PySyft was also used to create a model and training plan to solve the MNIST classification problem using federated learning.

The model plan was simply a MLP class describing 2 linear layers model with ReLU activation function. Then the functions of training plan were defined - set_params, cross_entropy_loss and sgd_step. Finally, a simple averaging plan was developed which is same as the one used in Flower framework. The averaging step takes all the model updates and finds the simple mean. Finally, the configuration of the server was defined.

4.4.2.2 Hosting on PyGrid using PySyft

After the creation of the model and the training plan, the PyGrid domain is started on the local machine and the plans are then hosted (stored) in the domain. PySyft basically allows developers to seed the PyGrid server with the base model and set rules which oversees the clients for the training process.

4.4.2.3 Training using PySyft

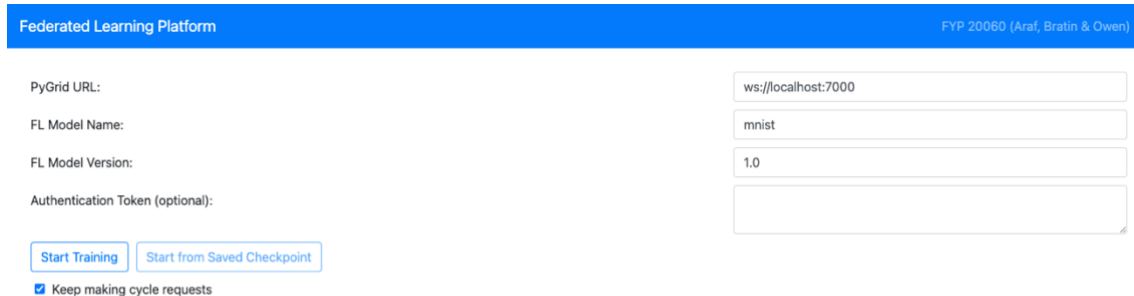
The hosted model can be accessed using the APIs provided by PyGrid. First, the client requests the server to be accepted into the training cycle. Once the client gets accepted, the model and the training plans are downloaded by the client (using jupyter notebook). It then uses the data stored locally to train the model. On the completion of the training process, the client sends back the updates to the PyGrid server. The server handles the aggregation as defined in the plans. This process continues until the client is rejected from the training cycle without timeout, indicating that the FL training is done. The validation accuracy obtained was ~80%.

4.4.3 Syft.js

Syft.js is a frontend PySyft worker library built on top of TensorFlow.js for FL which allows developers to perform federated learning on the web browser. It provides APIs to communicate with the PyGrid domain and run the plans specified by PySyft Plans in a browser. It is an alternative to training using PySyft in jupyter notebook as mentioned in Section 4.4.2.3.

In Syft.js, a worker object is created which is responsible for starting a job, i.e., the training process. Like training on jupyter notebook, the worker requests for a training cycle. Once the worker is accepted, it downloads the model and the plans

and starts training locally. Syft.js seamlessly integrates with the PyGrid API. It also supports inference of PySyft ML models which can be written in either PyTorch or TensorFlow.



The screenshot shows the 'Federated Learning Platform' interface. At the top, there is a blue header with the text 'Federated Learning Platform' on the left and 'FYP 20060 (Araf, Bratin & Owen)' on the right. Below the header, there are four input fields for configuration: 'PyGrid URL:' with the value 'ws://localhost:7000', 'FL Model Name:' with the value 'mnist', 'FL Model Version:' with the value '1.0', and 'Authentication Token (optional):' which is empty. Below these fields are two buttons: 'Start Training' and 'Start from Saved Checkpoint'. At the bottom left, there is a checked checkbox labeled 'Keep making cycle requests'.

Figure 4.23 Frontend design of the web platform

During the entire process, the data stays on the user’s device which is the desired feature of a FL platform. It is to be noted that the security protocol of FL such as Secure Multi-Party Computation and secure aggregation using peer-to-peer WebRTC connections is still in progress, but they can be manually implemented to ensure zero data leakage.

4.4.4 Limitations

The Syft.js framework is new and hence it’s not robust. We have developed the frontend (using Bootstrap as the frontend library) and managed to connect the web browser worker to the PyGrid domain running locally on the computer. The issue our team faced was that the training plans were not being downloaded by the worker which resulted in an error. This caused the training process to not begin. we suspect the problem might be due to the incompatible versions of PyGrid and Syft.js as this issue was not encountered during the training process conducted on jupyter notebook. But it is to be noted that once the training is completed on the jupyter notebook, the web browser recognized that and returns the status ‘training complete’. In the future, once the PySyft version ‘0.5.0rc1’ becomes compatible

with Syft.js, it will be able to communicate with the web browser with giving any errors.

4.5 Sherpa AI

Sherpa AI is the last framework we researched on that allows FL and was developed to facilitate open research. It aims to support 100% of the ML algorithms used in the industry while preserving data privacy. Similar to TFF, this framework is mainly used to simulate FL scenarios with custom models, algorithms, and data. It leverages Differential Privacy to ensure the security of data. The training process is done on each node to build a global shared model.

The framework was experimented with the MNIST dataset.

We performed two sets of experiments, first, where the total number of data was kept constant, and second, where the number of data samples per client was kept constant.

No. of nodes (clients)	No. of data per client	Global model test accuracy	Training Time(s)
1	3000	0.8541	74.85
5	~600	0.8371	206.66
10	~300	0.7858	386.08
15	~200	0.7495	461.30
20	~150	0.5730	721.09
25	~120	0.6344	903.46
30	~100	0.5525	960.65

Table 4.7 Results of simulation with constant number of data

Table 4.7 shows the result of the simulation keeping the total number of data constant (3,000 data samples). The training was run for 3 rounds. As expected, the accuracy of the global model gradually decreases, and the training time gradually increases as the number of clients is increased. The data gets scattered over several clients and the local training is only able to see limited data points leading to lower accuracy.

No. of nodes (clients)	Total no. of data	Global model test Accuracy	Training Time(s)
1	~300	0.6351	63.41
5	~1500	0.6812	195.66
10	~3000	0.7501	359.30
15	~4500	0.7785	604.89
20	~6000	0.7837	727.94
25	~7500	0.7777	865.50
30	~9000	0.7930	934.86

Table 4.8 Results of simulation with constant data samples per client

Table 4.8 shows the results of the simulation keeping the data samples per client constant (300 data samples). The training again was run for 3 rounds. The accuracy and the training time gradually increase as the number of clients is increased which is the expected result as the model get to see more and more data points which improves the overall generalization capability of the global model.

In the future, other datasets such as CIFAR10 and Cats_vs_dogs can be tested on this platform. The accuracy obtained through simulations can then be used as the baseline for the FL platforms for comparison.

In conclusion, this section described the products, experiments, and results of our final year project. We performed extensive research on four frameworks and managed to deliver three solutions - CLI platform (Flower framework), Web FL platform (PySyft, PyGrid and Syft.js) and a 1-1 private ML platform (PySyft - Duet).

5 Difficulties and Limitations

The following section describes the general difficulties and limitations faced during the building and testing of the FL platforms.

5.1 Age of FL Technology

One of the limitations that this project faced was that the technology driving Federated Learning is new. The inception of the concept happened in 2016. Therefore, there were limited resources to learn from and solve issues. Moreover, some obstacles that the project faced like, incompatible versions of Syft.js and PySyft cannot be easily solved as it requires deeper understanding of the field that is way beyond our group's capabilities.

5.2 Deployment of FL Platform

As mentioned in Section 4, frameworks like TFF, Flower and Sherpa AI, are in their early stage of development. It would require few years until these frameworks reach maturity and are capable of being deployed at production level. Currently, they only support local simulations and hence are suitable for research purposes only. Most of the frameworks were thus utilized to perform experiments to gain better understanding of not only the frameworks at hand but FL in general.

5.3 Complexity of Security Protocols

The security protocols of FL are constantly being research and improved. But recent research has shown that there might be potential security breaches during the training process if the platform uses Generative Adversarial Network [\[2\]](#). Hence, in few cases, such as the one mentioned before, it is impossible to ensure data privacy for the clients holding the data.

5.4 Communication Speed

One of the testing requirements of the federated learning platform is efficiency. The communication speed between the participants and the server will be noted as the number of users increase. While testing the efficiency of the deployed platform, one needs to make sure that the internet speed is constant as it might lead to varying results making the tests invalid. Add to that, slow internet connections can become bottlenecks as the packets of data containing the encrypted gradients might not even reach the server, causing frequent timeouts. Thus, finding a suitable internet connection might prove to be difficult.

Another point to note is that the clients and the server will be constantly communicating with each other. Consequently, high communication cost will be incurred for training and testing the ML models.

6 Future Work

In this report, after tedious research, we have suggested platforms for performing FL. But the platforms are yet to be deployed due to the limitations posed by the respective frameworks. Once the FL platforms are capable of being deployed in the future, they will be then compared on their accuracy, efficiency, and privacy. The model, trained on these platforms, will also be compared with to the ones trained using traditional machine learning method, where the data is collected from all the clients on a single system and then training is done on the same system. Add to that, the privacy of data needs to be confirmed before it is ready to be used by the hospital to train the ML model for detecting COVID-19.

7 Conclusion

In this report, the issues behind traditional ML process were discussed, where all the data needs to be centralized and data privacy was pointed out as one of the major concerns. This acted as the motivation for our project which aims to build a Horizontal FL platform where participants can collaboratively train a global ML model without fearing the privacy of the data. We performed rigorous research and evaluation on different federated learning frameworks and ultimately presented few platforms that we hope will help AI InnoBio to train their COVID-19 detection model using the data of hospitals in the Asia-Pacific region. Finally, we discussed the difficulties faced in this project and the future prospect of our work.

In conclusion, federated learning has the capability to transcend the limitations faced by the Machine Learning community. With proper implementation and vigorous testing, it has the potential to revolutionize many industries.

References

- [1] B. Han, "An Overview of Federated Learning", *Medium*, 2020. [Online]. Available: <https://medium.com/datadriveninvestor/an-overview-of-federated-learning-8a1a62b0600d>. [Accessed: 27- Oct- 2020].
- [2] Q. Yang, Y. Liu, T. Chen and Y. Tong, "Federated Machine Learning: Concept and Applications", *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1-19, 2019. Available: 10.1145/3298981.
- [3] J. Konecny, B. McMahan and D. Ramage, "Federated Optimization: Distributed Optimization Beyond the Datacenter", 2015. Available: <https://arxiv.org/abs/1511.03575>. [Accessed 27 October 2020].
- [4] A. Hard et al., "Federated Learning for Mobile Keyboard Prediction", 2018. Available: <https://arxiv.org/abs/1811.03604>. [Accessed 27 October 2020].
- [5] B. McMahan and D. Ramage, "Federated Learning: Collaborative Machine Learning without Centralized Training Data", *Google AI Blog*, 2017. [Online]. Available: <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>. [Accessed: 27- Oct- 2020].
- [6] Reuters, 2020. Israeli Hospital Trials Super-Quick Saliva Test For COVID-19. [online] U.S. News & World Report. Available at: <<https://www.usnews.com/news/top-news/articles/2020-08-13/israeli-hospital-trials-super-quick-saliva-test-for-covid-19>> [Accessed 7 October 2020].
- [7] D. Beutel, T. Topal, A. Mathur, X. Qiu, N. Lane and T. Parcollet, "Flower: A Friendly Federated Learning Research Framework", 2020. Available: <https://arxiv.org/pdf/2007.14390.pdf>. [Accessed 11 January 2021].