THE UNIVERSITY OF HONG KONG
DEPARTMENT OF COMPUTER SCIENCE

Final Year Project Individual Final Report

Federated Learning Platform for Covid-19 Detection

Khan Khondoker Araf Hasan 3035477446

Teammates
Kwan Pok Man 3035477173
Ghosh Bratin 3035437692

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

April 18, 2021

Abstract

Advancement in technology has made it feasible for machines to imitate humans to perform complicated tasks by learning from experience and adjusting to new inputs which are generally referred to as Artificial Intelligence or Machine Learning. The traditional machine learning method required all the training data to be centralized in one location.

With the modern era of big data, countries or organizations are paying more attention to data privacy and have formulated strict privacy provisions. This creates a situation where the integration of data is difficult, making it impossible for different organizations to share data to train a shared effective machine learning model for better accuracy.

Federated Learning is a new concept that arose in 2016, that hopes to tackle the aforementioned issue. The project aims to incorporate this new technology and create a platform where machine learning could be achieved with data privacy as its main consideration.

The core part of this project is to design 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.

Research and evaluation on different Federated Learning methods in Tensorflow federated has been done. Performance of Federated Learning are also done in Sherpa AI Federated Learning framework. Moreover, a basic client-server communication Command Line Interface platform using Python Flower framework has been developed. Development of the Federated Learning platform using Duet has been done. A platform using Sftty.js is in the preliminary stage and need more improvements.

Acknowledgement

Firstly, I would like to express my sincere gratitude to Dr. S.M. Yiu from the Department of Computer Science. This project was not originally listed on the final year project list and I am grateful that he invited our group to work on this project which involves a brand new concept in the computer science field.

Having such experience would be hugely beneficial to my career as a software engineer, especially in the financial field where it values data privacy tremendously.

Lastly, I am truly thankful to be able to work with, Owen Kwan and Bratin Ghosh, my teammates who supported each other and helped me a lot throughout the project. I am looking forward to continuing to work with them for the remaining progress.

Contents

List of Figures

List of Tables

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

1 Introduction

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 machines that can learn and act on their own will 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 with a large number of data sets. Well known and frequently used technologies such as YouTube recommendation algorithms, Google search engines, 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 many industries or countries, there is only limited data, or the data existing are of poor quality. In other cases, quality data exist as isolated islands on different edge devices such as mobile phones and personal computers across the globe. Would it be possible to transport all the data across different organizations and break the barriers between different data sources? In fact, it is very difficult. Firstly, the cost of transportation and integration of huge data would come at an enormous cost. Most importantly, the extraction of such data is prohibited and restricted by strict privacy-preserving laws either enforced by corporations or governments. Even within the same organization, data integration would face heavy resistance due to administrative issues and privacy security. The realization of ML in different fields is far more difficult than imagined.

1.2 Federated Learning

The challenge created by privacy concerns gave rise to a new concept proposed by Google in 2016 called Federated Learning [1]. FL provides a brand-new way for

ML models to collect the data required to train effectively. While traditional ML requires a central server to collect all the data from different local devices and perform centralized processing and training, FL performs ML model training at the local device itself using its local data.

1.2.1 Three types of Federated Learning

This subchapter introduces the three types of Federated Learning in detail. According to the type of data, FL can be classified into horizontal federated learning, vertical federated learning, and federated transfer learning.

1.2.1.1 Horizontal Federated Learning

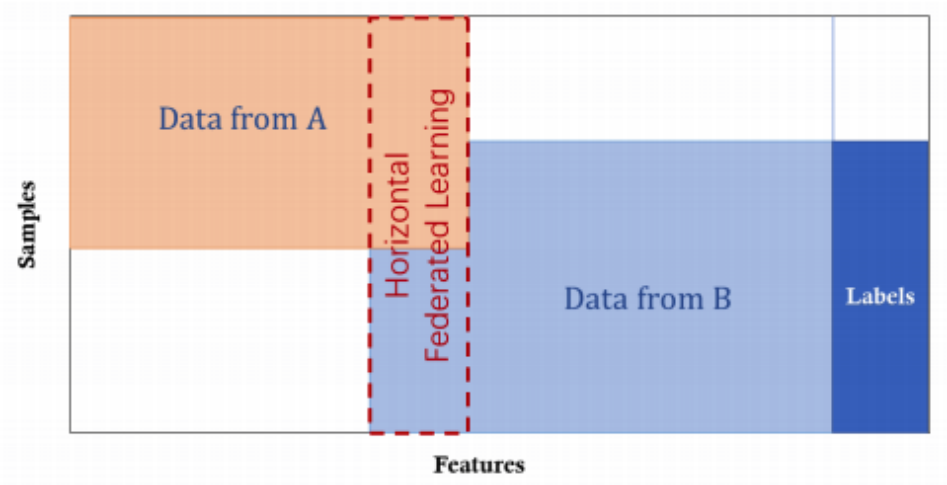

Figure 1.1 *Horizontal Federated Learning [2]*

Horizontal Federated Learning is also known as sample-based federated learning. It is designed for data sets that have the same feature space but different in the sample [2]. Given two separate universities which have different students in their database. Although the user group is different and the intersection set between the group is very small, universities store similar types of information because they are both in the education sector, hence, the feature space is the same. In this case, horizontal FL would be used.

1.2.1.2 Vertical Federated Learning

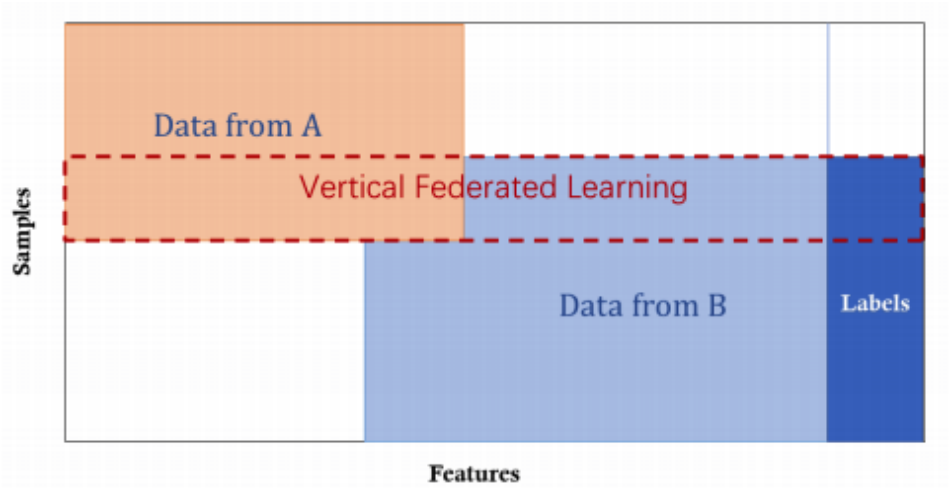

Figure 1.2 *Vertical Federated Learning [2]*

Vertical Federated Learning, or Feature-based Federated Learning, is opposite of Horizontal Federated Learning. Instead of having the same feature space and different sample space, Vertical Federated Learning is designed for data sets that share the same sample space but differ in feature space. Consider a bank and an e-commerce company located in the same city, they are likely to contain the same users in their system, the intersection of user space is large meaning they have similar sample space. However, both of their businesses are different and will contain different sets of data. For example, the bank records the user's credit rating and income while the e-commerce records the user's purchasing history, it is to say that their feature space is vastly different. Vertical FL will be used for a situation where a machine learning model requires information both the companies retain, such as building a prediction model for product purchase based on a user’s credit rating and income.

1.2.1.3 Federated Transfer Learning

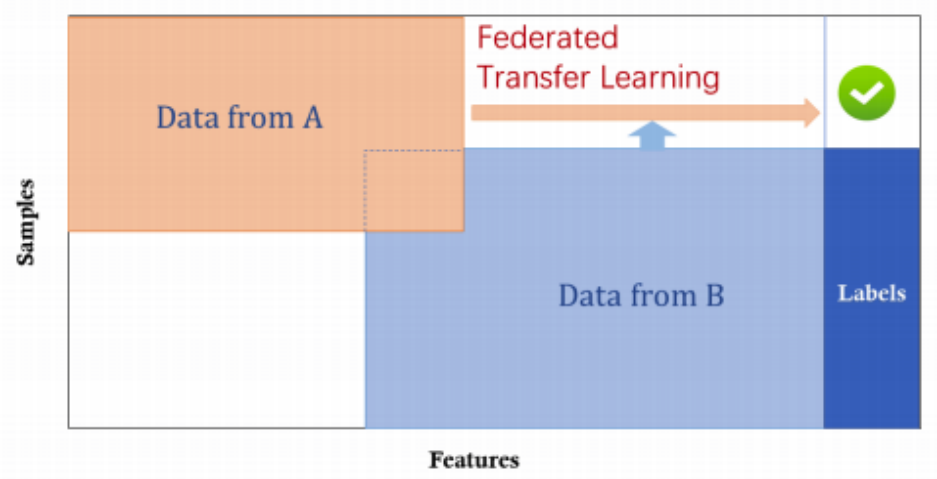

Figure 1.3 *Federated Transfer Learning [2]*

Federated Transfer Learning is the most complicated among the three types of federated learning models. It is used in complex scenarios where data sets differ both in sample space and feature space. For example, consider one bank located in Hong Kong and an e-commerce company located in London. Due to geographical location, the users of both companies are vastly different, the intersection of the user group is small meaning the sample space is different. On the other hand, both companies operate different business models and contain a different type of information about the user meaning the feature space are also different. In this situation, transfer learning techniques need to be applied in order to allow the sample space and feature space to be used under FL [2]. A common representation needs to be established with both the feature space using the limited intersection of sample space which will be later 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.

1.2.2 Benefits of Federated Learning

FL allows devices like mobile phones to collectively learn a shared ML model while keeping the training data within the device which was previously required to be sent to the centralized server in the traditional method [5]. This approach will solve the issue of data privacy and enforce strong security. This allows organizations such as hospitals or banks to perform ML under reduced liability.

FL enables real-time operation since the model resides with the devices itself. FL reduces the time cost of data transmitting between devices and centralized servers as the transmission involves model updates only instead of raw input data which are significantly smaller in size. Since the model resides within devices in FL, internet connectivity is no longer needed for ML operation under FL.

One might ask the question of whether edge devices are able to handle complex computation involved in ML in a timely manner? In FL, the amount of hardware infrastructure required is reduced where only minimal hardware is needed, and a normal mobile device would be able to handle FL [3].

1.2.3 Challenges of Federated Learning

Communication is a core issue in FL Architecture. An effective communication method is crucial to improve the efficiency of the system by reducing the total number of communication rounds and also send small iterative updates as part of the training process.

FL systems should also be able to handle a low device participation rate where only a small fraction of the total devices is being active at any given moment. In most cases, participating devices are not standardized, the FL system should also be able to tolerate variability of the hardware, including storage, computational as well as communication capabilities.

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

One of the prominent related work of Federated Learning in the industry is done by Google, the company that coined the term Federated Learning. They are actively testing Horizontal FL in Gboard on Android, the Google Keyboard [2].

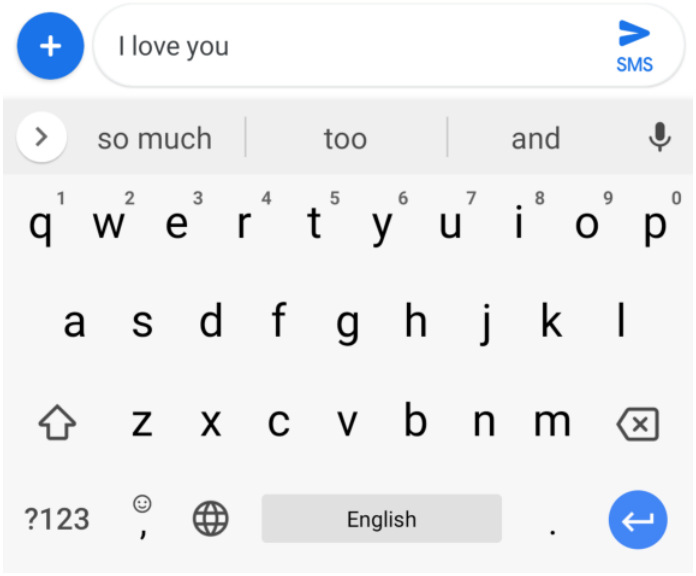

Figure 1.4 *Gboard next word prediction in Gboard [4]*

When Gboard suggests a word query as demonstrated in figure 1.5, the device will locally store the information regarding the current context and whether the suggestion was used. FL processes all the history on-device to suggest improvements to 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 six chapters. The first chapter introduces the concept of Federated Learning, the three types of FL, the benefits of using FL, the challenges behind FL, and a previous work of FL done by google with GBoard.

Chapter two presents the objective of the project and also outlines the significance of this project.

Chapter three goes through the details of implementing a FL platform. It will first go through the reason why Horizontal FL is chosen. Then, the architecture for horizontal FL will be explained followed by the security protocols that are known for horizontal FL.

Chapter four presents the current progress of the project.

Chapter five mentions the future work, the schedule, and the challenges that might be encountered throughout the project.

Chapter six concludes the report. It wraps up the motivation behind this project and the progress so far.

2 Objectives and Motivation

2.1 Project Objectives

This project aims 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.

After the platform is built, it will be evaluated under three main criteria – accuracy, efficiency, and privacy. For accuracy, the platform's accuracy will be compared with that of a model trained under a centralized location, or so called the traditional machine learning method. In the efficiency aspect, the platform will be evaluated by the speed at which it trains a model with a different number of edge devices and the communication cycle that it requires. Different privacy protocols will be researched and tried to find the best fit for the best platform that is most secure.

2.2 Project Motivation

AI InnoBio Limited is a Biotechnology Start-up based in Hong Kong that utilizes an industry-leading CMOS sensor technology to develop a novel hand-held spectrometer device. This device has the capability to incorporate artificial intelligence to conduct saliva tests to determine within a second whether a certain patient is infected with the novel coronavirus. A hospital in Israeli had conducted several clinical trials with hundreds of patients with this new artificial intelligence-based device and is able to achieve a 95% success rate of identifying evidence of the virus in the body [6].

Such technology has not come into sight in Asia yet and AI InnoBio hopes to expand its business to allow different countries in Asia to use the device to perform a fast, accurate, and low-cost Covid-19 detection test. Some regions such as Hong Kong do not have enough sample data to perform effective machine training. In order to obtain a better machine learning model, it is crucial to obtain more test data across different countries. This is where the problem of data privacy comes into play, different countries and hospitals will not be willing or under heavy restriction to share the test results across different parties. Therefore it is difficult to use the traditional machine learning method where sufficient data sets need to

be centralized to train the model. Therefore, a platform is needed to perform federated learning with different clients to improve the accuracy of the model while keeping the data privacy intact.

This project aims to provide groundwork for the FL platform that is required by the company where it will aid the testing of Covid-19 especially in countries that have huge numbers of to be confirmed cases. Although the platform is our ultimate goal, the other major goal of the project is to research federated learning and conduct research of different 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

The architecture of the three types of Federated Learning is vastly different since each of them is dealing with the different nature of data sets. This chapter first explains why Horizontal FL is used in this project, followed by a presentation of the architecture of the Horizontal Federated Learning platform that this project is aiming to build. It is the most common type of federated learning system in the market compare to the other two types, vertical and transfer learning [2].

3.2 Rationale for using Horizontal Federated Learning

This platform will be built with the consideration of the Covid-19 detection machine learning model developed by AI InnoBio. However, the machine learning model is not developed yet and speculation needs to be made for the field that the machine learning model and the data sets to be used. Since the detection method is the same, which is the spectrometry test done on the CMOS sensor device, the feature space will be the same. However, the data are coming from different regions, so the sample space is not the same. Therefore, Horizontal FL is the best type of FL to be used in this case.

3.3 Horizontal Federated Learning Architecture

This project will adopt the architecture for a horizontal FL system shown in Figure 2.1. This architecture is very similar to what section 1.2.1 has mentioned and this subsection will delve deeper into the technical details on how each operation works.

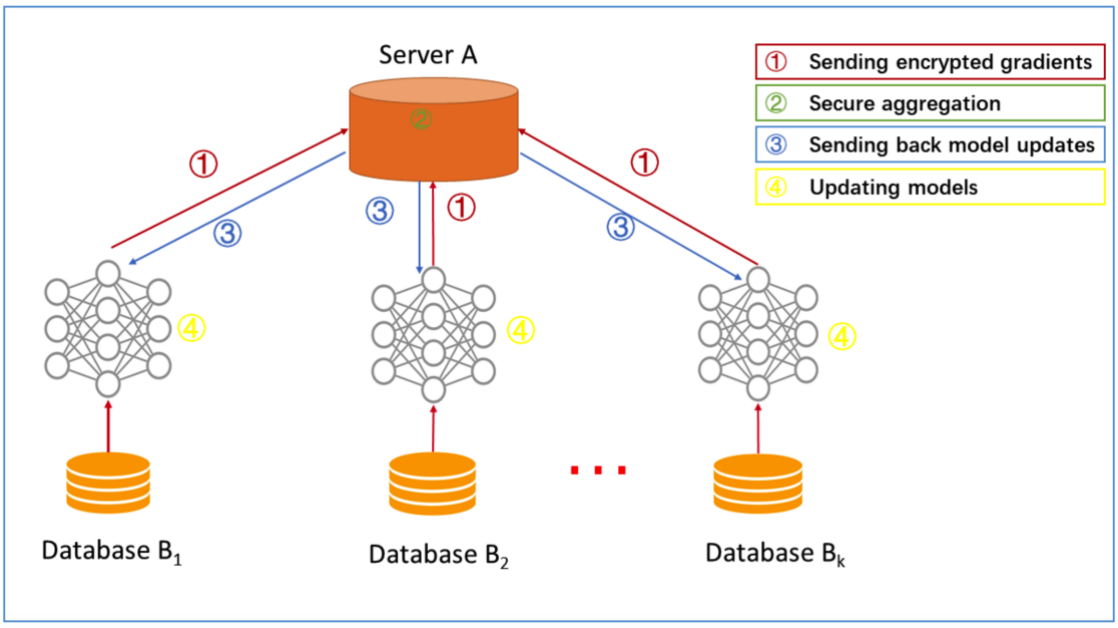

Figure 3.1 *Architecture of Horizontal Federated Learning System [2]*

In this system, k number participants as shown as Database B1 to Bk contains the same data structure, same feature in the database, and will collaboratively train the machine learning model with the aid of the centralized cloud server, server A in Figure 2.1. There are four steps on how a machine learning model gets trained in the federated system. The first step involves participants computing the training gradients locally, and mask a selection of gradients secured with encryption, differential privacy techniques, then send the masked result to the server. Next, the cloud server will pick up encrypted gradients or weights from different participants and perform secure aggregation without learning any information about any participant in the process. Step three is where the server sends back the encrypted aggregated result to the participants. Lastly, the participants will decrypt the result of the gradient and update their respective ML models.

The above learning cycle will continue indefinitely until the loss function converges, which means the deviation of the ML model predictions from the actual results is within an acceptable range.

It should be noted that this architecture is independent of any specific machine learning algorithms such as logistic regression or DNN, the architecture focuses on the sharing of model parameters for training purpose only.

3.4 Security Protocol

Privacy is essential to Federated Learning and is the reason why Federated Learning came to light. Security models and protocol is crucial to provide privacy guarantees. A Horizontal Federated Learning system assumes that the participants are honest and security measures are only done on the server which is honest but curious [2]. This is to say, only the cloud server can compromise the data privacy of participants. This subsection will briefly go through three different popular security protocols, which are Secure Multi-party Computation, Differential Privacy, and Homomorphic Encryption.

3.4.1 Secure Multi-party Computation

The name Secure Multi-party Computation implies this protocol involves multiple parties. It provides a framework where participants know nothing except the input and output of the FL system which ensure zero-knowledge which is desirable [2]. However, the zero-knowledge property requires complex computation and may not be achieved in this project efficiently. In certain scenarios including this project, partial knowledge disclosure is considered acceptable when security guarantees are promised because of that.

3.4.2 Differential Privacy

Another protocol is that of Differential Privacy. It is a protocol where generalization methods are used to obscure sensitive fields until outsiders are not able to distinguish which participants are the data referring to, or in other words, noise is added to the data. This makes the data impossible to recover to protect user privacy. However, this protocol of adding noise to data might affect the accuracy of the data and the ML model [2].

3.4.3 Homomorphic Encryption

Homomorphic Encryption used an encryption mechanism to protect data as the name implies. It encrypts the parameter exchange during the ML progress. Different from Differential privacy, data, and model is not being transmitted.

3.5 Summary

This area is not the priority of this project. However, research will be conducted to see which protocol is the best fit for the platform using the 3 metric of success -- accuracy, efficiency, and privacy. We will then try to implement one of them to the platform. However, it might not be possible because it's still a new technology, therefore might not be implementable yet, or it may require complex computation or algorithms out of our capability.

This chapter explains the reason why this project is using Horizontal Federated Learning and lays the architecture behind building a Horizontal FL system. The three security protocols that are applicable to the system are also described in this chapter. The next chapter will report the project’s current progress.

4 Experiments and Results

4.1 Overview

This chapter presents the work that have done in the project. The project consists of 4 parts. One part is the research and evaluation on different federated learning methods using Tensorflow Federated which will be outlined in subchapter 4.2. The second part is the performance analysis of Federated Learning using Sherpa AI in subchapter 4.3. The third part is the development of the Federated Learning Command Line interface Platform using Python Flower Framework which will be explained in subchapter 4.4. The last part is the develop of the Federated Learning framework in the PySfty, PyGrid, and Syft.js ecosystem.

4.2 Tensorflow Federated

Cats_vs_dogs, CIFAR 10, and Fashion MNIST were used as the datasets in Tensorflow Federated. CNN was used for the first two datasets and simple dense neural network for the last dataset.

Three methodologies were tried on each of the dataset which are traditional centralized ML method, Tensorflow Federated Learning API, and Tensorflow Federated Core respectively. Traditional centralized ML method serves as a baseline for comparison. For Tensorflow FL API, a Keras model is created and wrapped as tff.learning.Model. The tff.learning API, where it is treated mostly as a black box, will then handle the FL logics. For Tensorflow Federated Core, the aim is to build a custom Federated Averaging algorithm. Instead of using the tff.learning API as a black box, we implement the functions within the API. Four main components are in the Federated Averaging algorithm which includes server-to-client broadcast, client update, client-to-server upload, and server update.

The training results of the 3 methods on the 3 datasets are listed below.

4.2.1 Fashion MNIST

Fashion MNIST is a black and white dataset consisting of a training set of 60,000 examples and a test set of 10,000 examples. Below are some examples of images in the dataset.

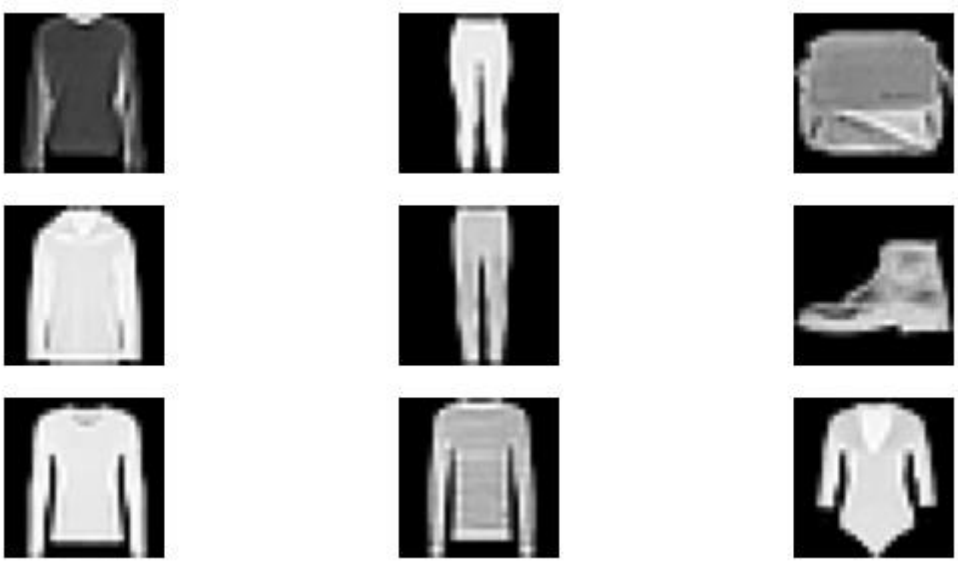

Figure 4.1 *Sample images from Fashion MNIST*

The images within the dataset are flattened and used it to train the fully connected neural network (FCNN) below.

```
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*

First, the network was trained with the traditional centralized ML technique using 10000 images. The validation accuracy reaches close to 84% after 10 epochs as shown in Figure 4.3.

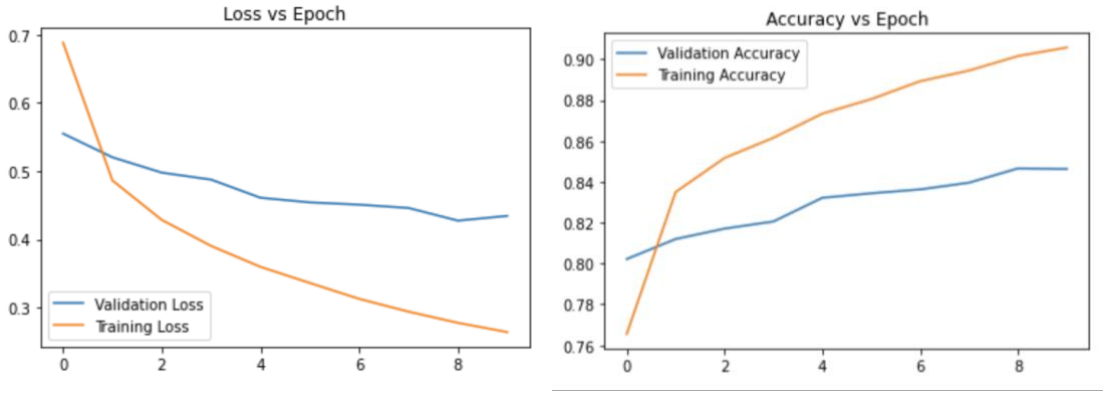

Figure 4.3 *Loss and accuracy of neural network trained with traditional machine learning technique on Fashion MNIST*

Then, Tensorflow Federated Learning API is then used in the dataset. The dataset is separated into 100 clients each containing 1000 image simulate FL environment with decentralized data. The distribution of the classes in the first 3 clients are shown in Figure 4.4.

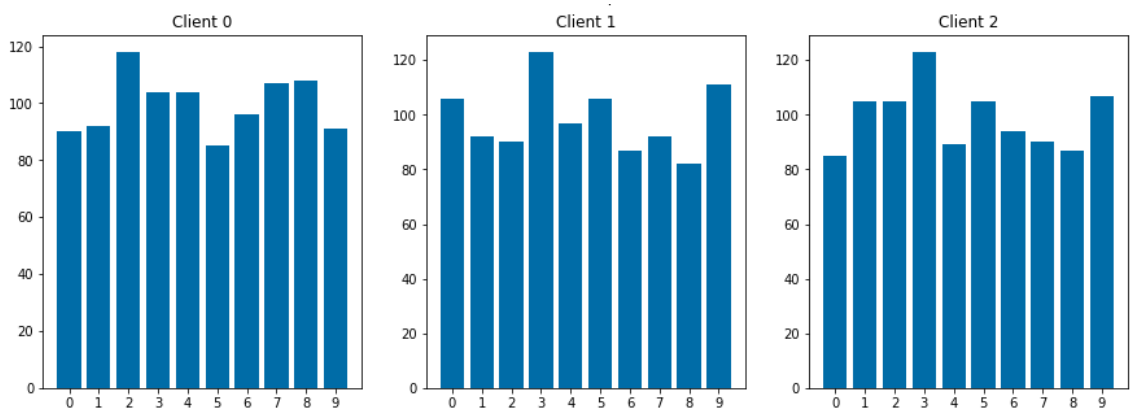

Figure 4.4 *Class distribution of Fashion MNIST*

Only the same 10 clients are selected for training at every round. As a result, the total number of image used is 10000. Figure 4.5 shows the validation accuracy with Tensorflow FL API.

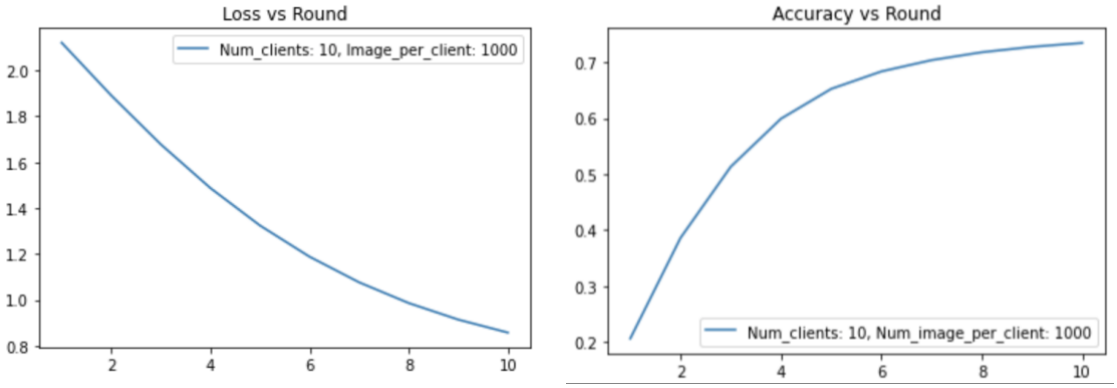

Figure 4.5 *Validation loss and accuracy of neural network trained with Tensorflow Federated Learning API on Fashion MNIST*

With the increase of the accuracy against rounds of training, we can see that Tensorflow FL API can perform ML without centralizing data from multiple clients. However, the accuracy after 10 rounds is only around 74 compared to that of Traditional ML method of 84% which indicate that FL although can work, the performance is slightly worst than traditional FL.

Lastly we trained the neural network by implementing the Federated Averaging algorithm using Tensorflow Federated Core. Figure 4.6 shows the validation accuracy.

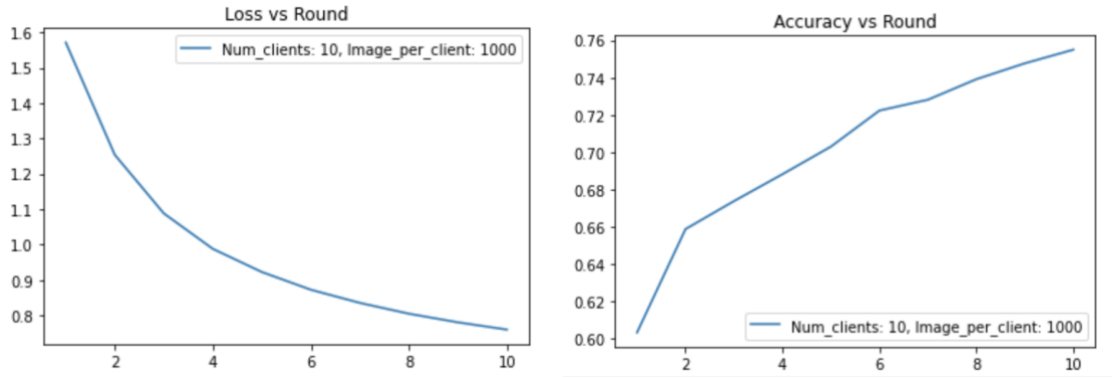

Figure 4.6 *Validation loss and accuracy of neural network trained with Tensorflow Federated Core on Fashion MNIST*

We can see the accuracy is close to the one that used Tensorflow FL API but still lower than that of Traditional ML method.

4.2.2 CIFAR10

CIFAR10 is a coloured dataset with 50000 training images and 10000 testing images. Figure 4.7 shows some of the images in the dataset.

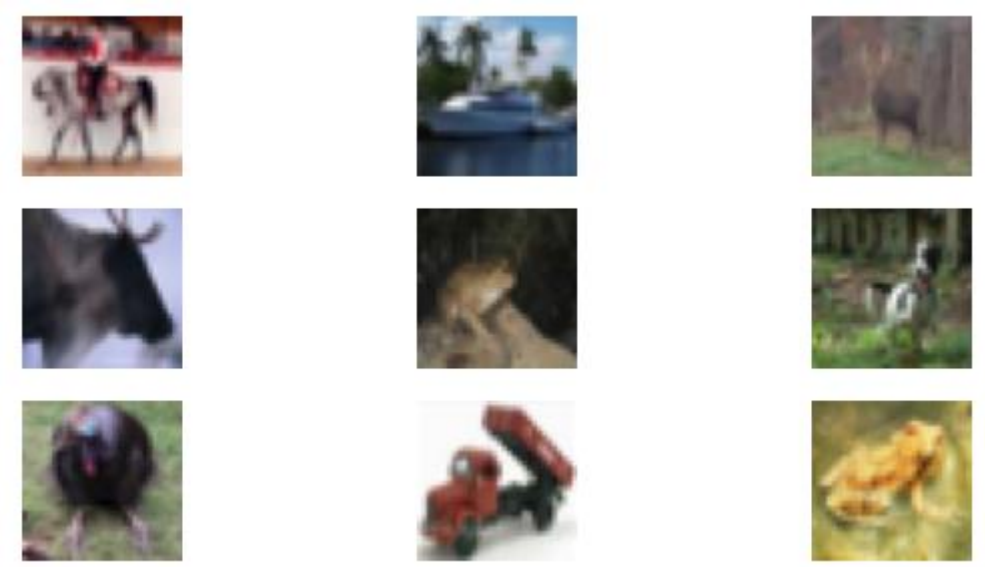

Figure 4.7 *Sample images from CIFAR10*

CNN is used to classify the images in CIFAR10. The model layout is 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 (MaxPooling2 (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 same methodologies in subchapter 4.2.1 are used in CIFAR10. Figure 4.9 and 4.10 shows the class distributions and the results from Traditional ML respectively.

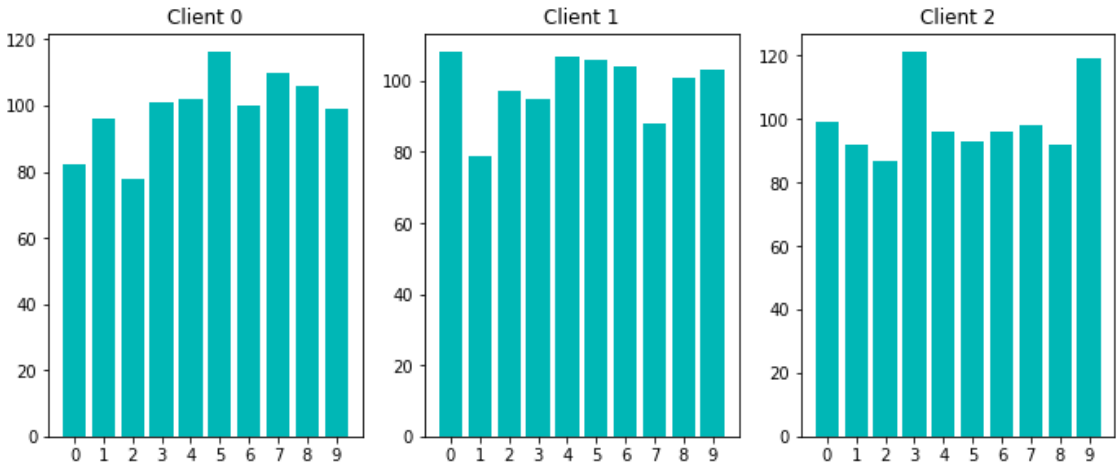

Figure 4.9 *Class distribution of CIFAR10*

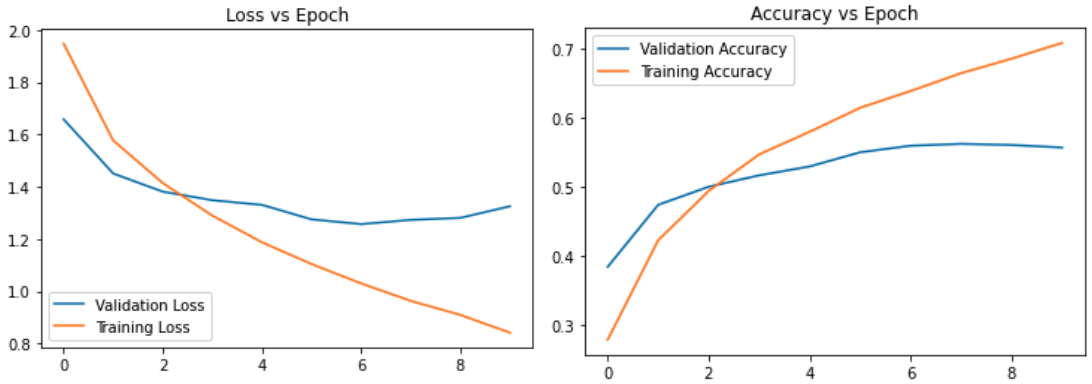

Figure 4.10 Loss and accuracy of neural network trained with traditional machine learning technique on CIFAR10

CIFAR10 is a more complex than Fashion MNIST because the images have 3 channels. Therefore, it explains the drop in validation accuracy after 10 epochs being much lower than that of the Fashion MNIST dataset.

Figure 4.11 shows the result for using Tensorflow FL API.

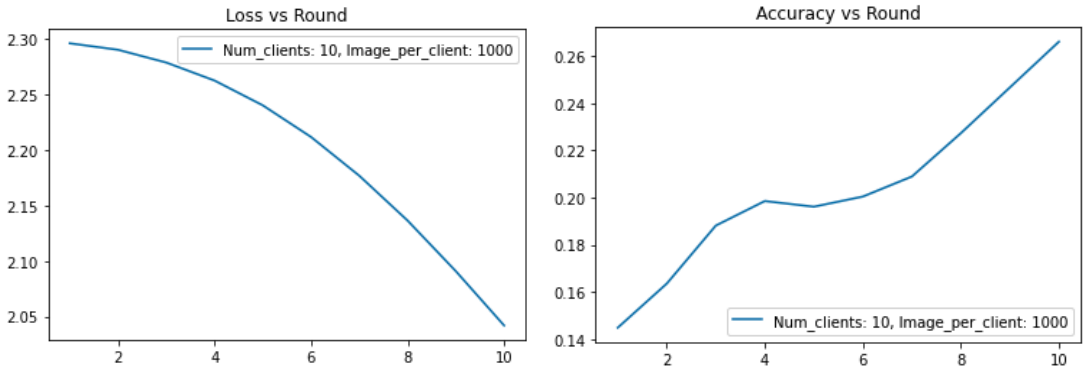

Figure 4.11 *Validation loss and accuracy of neural network trained with Tensorflow Federated Learning API on CIFAR10*

Figure 4.12 shows the result for using Tensorflow Federated Core.

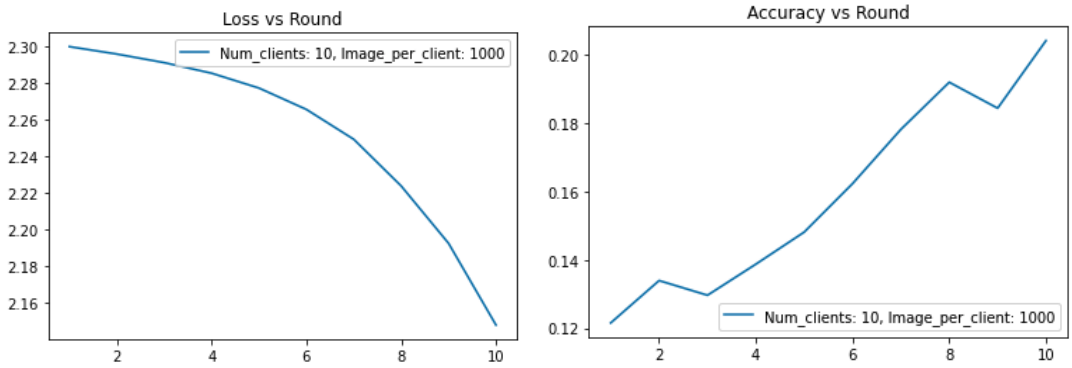

Figure 4.12 *Validation loss and accuracy of neural network trained with Tensorflow Federated Core on CIFAR10*

The accuracy for federated learning algorithm is much lower compare to that of using traditional machine learning for CIFAR10. It might be due to CNN in federated learning needs more rounds to converge, the image per client needs to be higher, or the number of clients participating the training is insufficient.

4.2.3 Cats_vs_dogs

Cats_vs_dogs is a coloured dataset of 25000 images. Figure 4.13 shows some examples of images in the dataset.

Figure 4.13 *Sample images from Cats_vs_dogs*

The same methodologies in subchapter 4.2.1 are used in *cats_vs_dogs*. However, in this case, each client only has 100 images. Figure 4.14 and 4.15 shows the class distributions and the results from Traditional ML respectively.

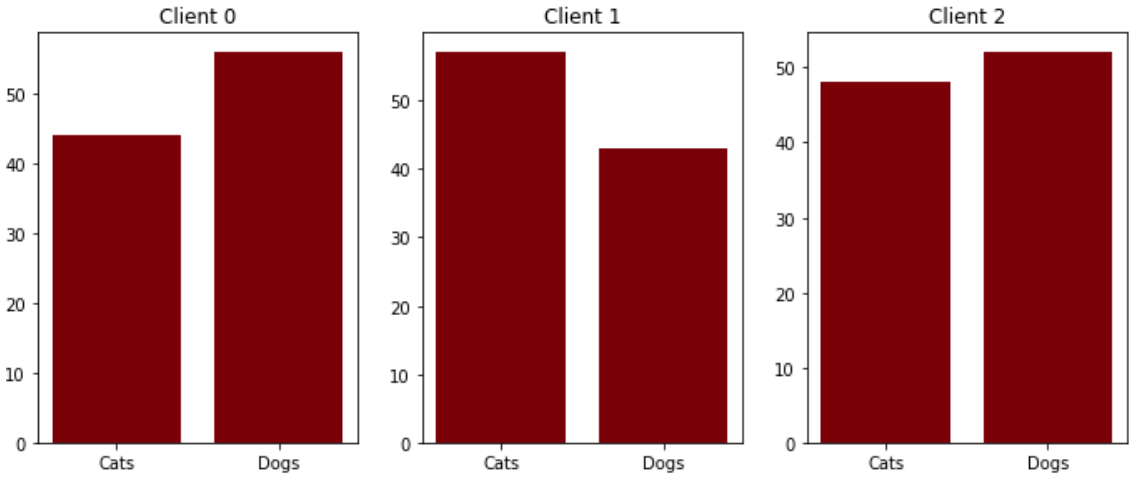

Figure 4.14 *Class distribution of Cats_vs_dogs*

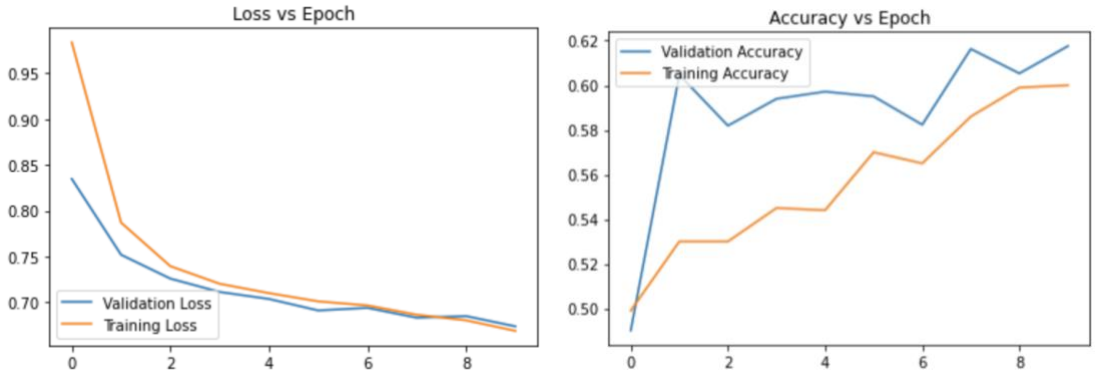

Figure 4.15 *Loss and accuracy of neural network trained with traditional machine learning technique on Cats_vs_dogs*

The cats_vs_dogs dataset is also a dataset with 3 channels, but there are only 2 classes, so the accuracy is in between CIFAR10 and Fashion MNIST.

Figure 4.16 shows the result for using Tensorflow FL API.

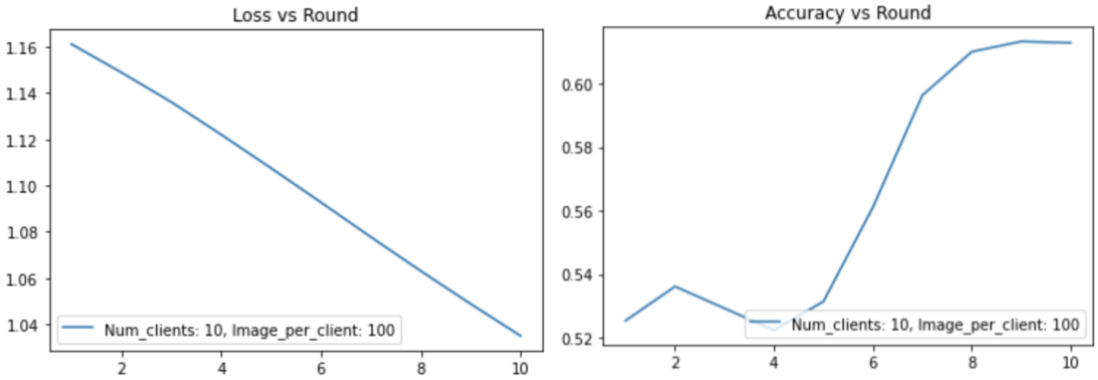

Figure 4.16 *Validation loss and accuracy of neural network trained with Tensorflow Federated Learning API on Cats_vs_dogs*

Figure 4.17 shows the result for using Tensorflow Federated Core.

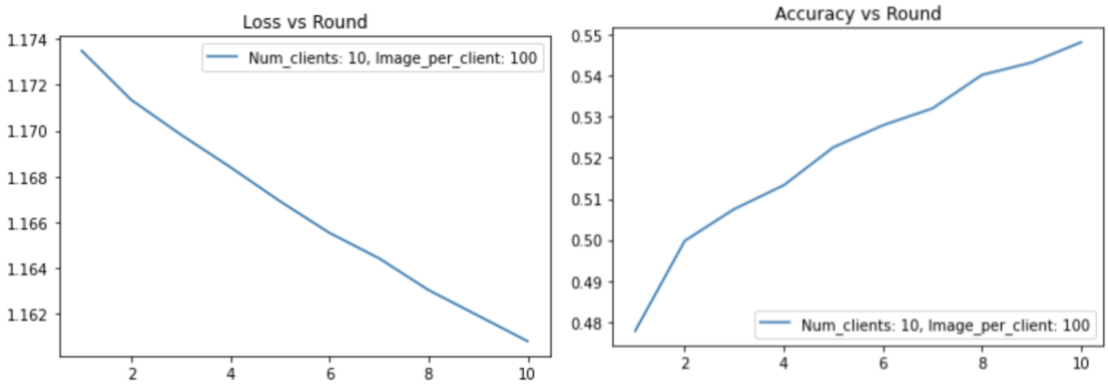

Figure 4.17 *Validation loss and accuracy of neural network trained with Tensorflow Federated Core on Cats_vs_dogs*

The accuracy using federated learning is slightly lower than that using traditional machine learning. However, the difference between the two is much smaller than the difference in CIFAR10. This might be due to the different number of classes in the two datasets, and the complexity of the datasets.

Accuracy of FL is lower than that of traditional ML as expected. However, since there is only two classes in cats_vs_dogs compare to 10 in CIFAR10. The difference between accuracy in FL and Traditional ML in cats_vs_dogs compare to that of CIFAR10 is smaller.

4.2.4 Experiments

More experiments are conducted to understand more about FL. In previous trainings, only the same 10 clients were used. We want to inspect what will happen if different clients are used in each round. From Figure 4.19, the accuracy of using different clients rise faster than that trained using the same clients. The final accuracy of training with different clients after 20 rounds is also slightly higher than using same clients.

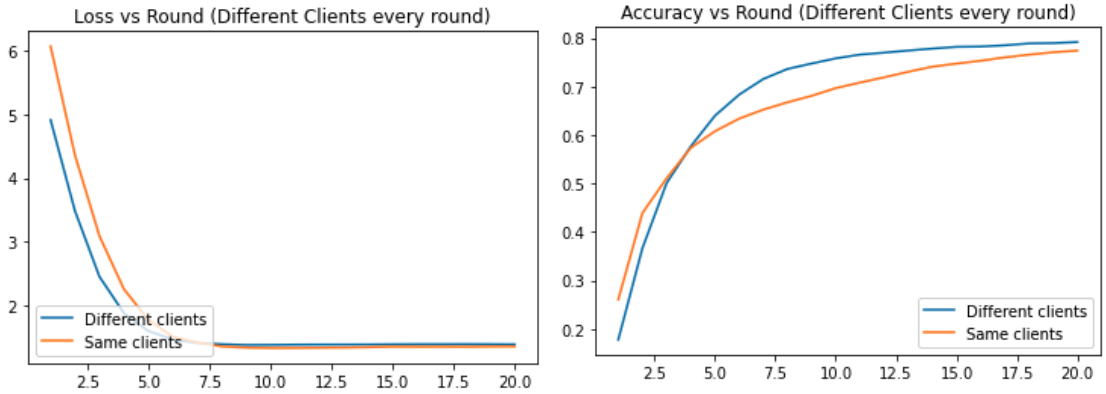

Figure 4.18 *Validation loss and accuracy of neural network trained with the same/different clients for each round*

Grid search is used to examine the performance of the algorithm with different number of clients, and different number of images per client. Figure 4.19 shows the result of different combination of clients and number of data per client.

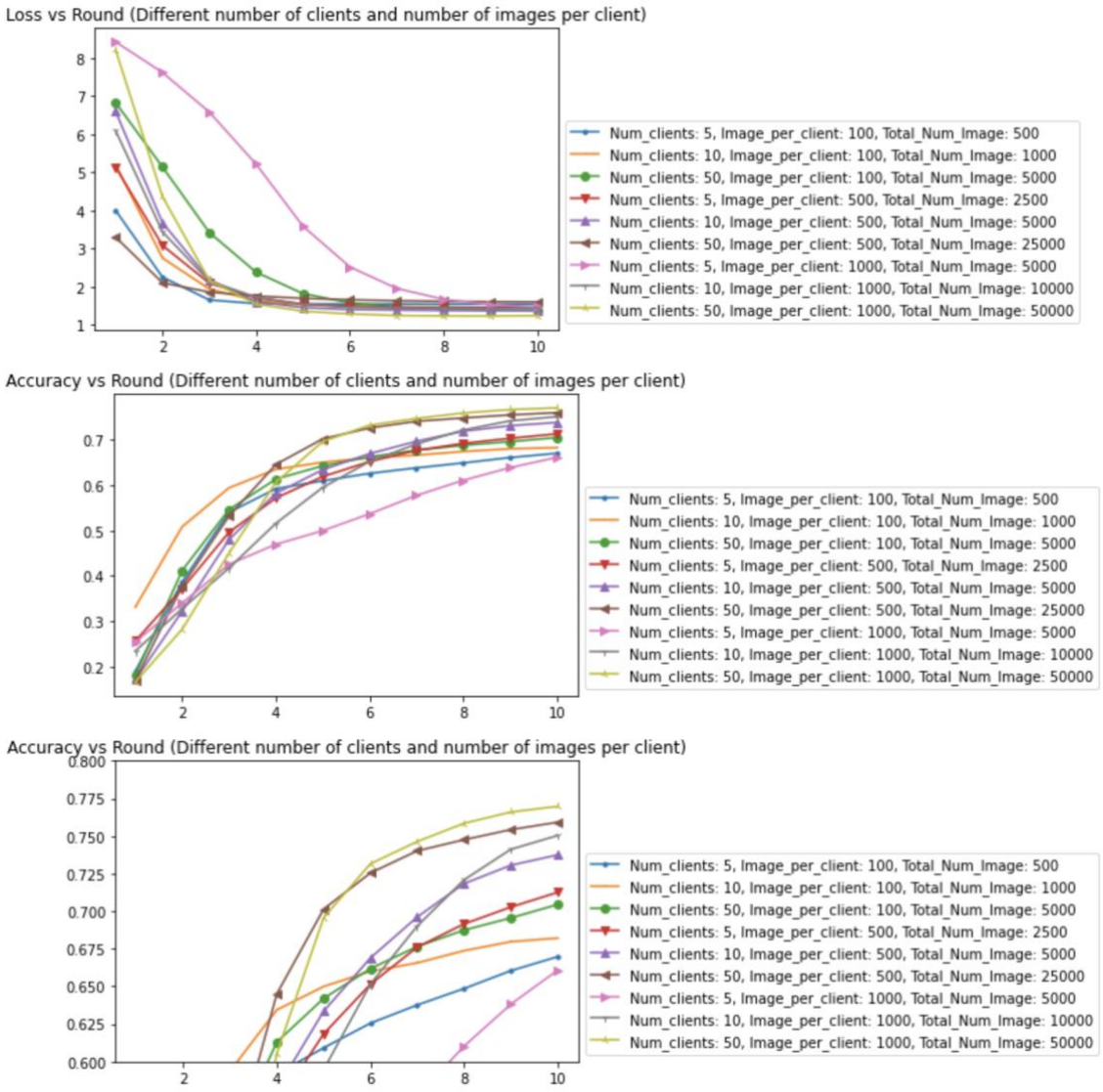

Figure 4.19 *Validation loss and accuracy of neural network trained with different number of clients and number of images per clients*

We can see that in general, the accuracy increases with the total number of images used in the training. However, an interesting finding is that when the number of clients is less, for example 5 clients, the model will perform worst when clients have more data. It might be due to too many data in the client causes overfitting. However, when there is more clients performing, the effect of overfitting will be cancelled.

Figure 4.20 shows the result of fixing the number of images used in the training to 5000.

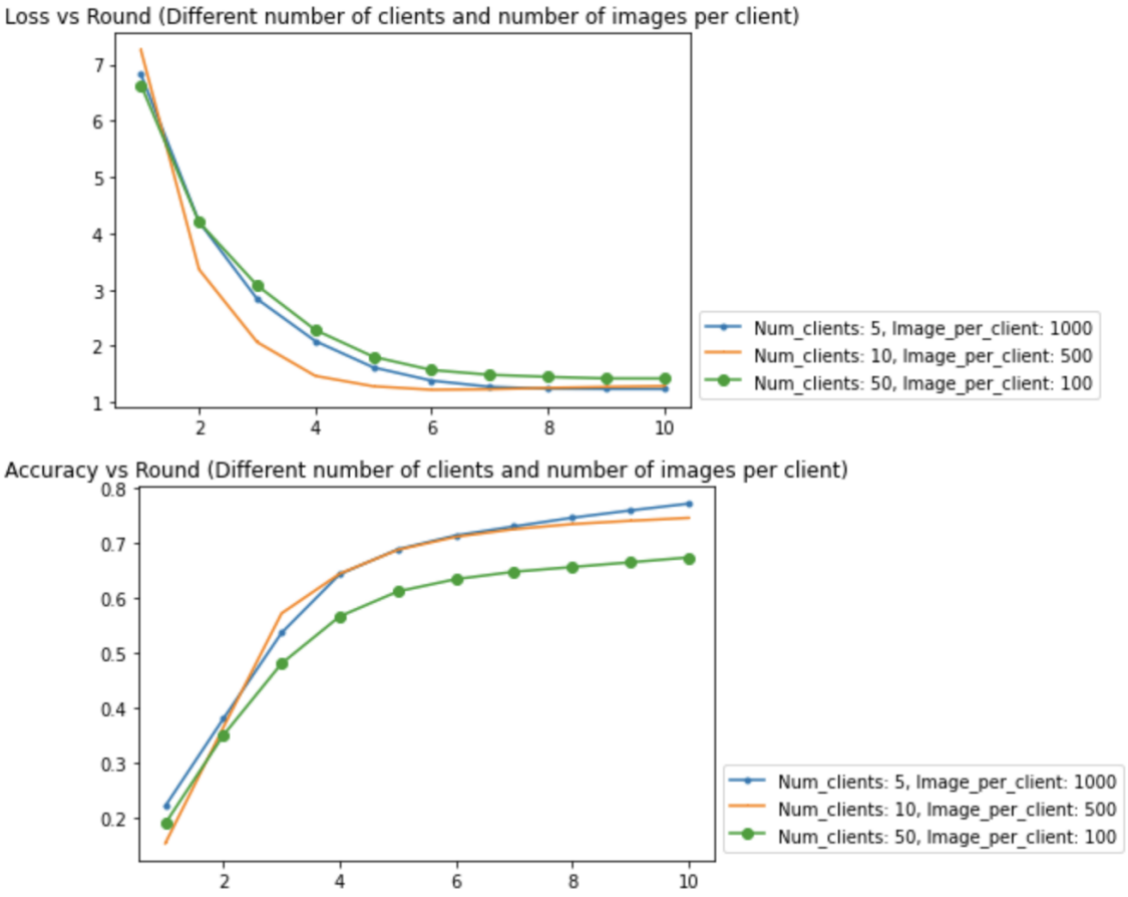

Figure 4.20 *Validation loss and accuracy of neural network trained with 5000 images in total*

Since in previous experiment we can see the accuracy for Traditional ML is better than FL. When there is less client, data is less decentralized, the FL model will perform better with a constant amount of total data.

4.3 Sherpa AI

Sherpa.ai Federated Learning and Differential Privacy Framework is developed by Sherpa.ai to facilitate open research on the field of FL. It serves to build models that can learn from decentralized data. This framework allows training the model locally on each node (client) to build a global model.

We choose this framework to research on the performance of Federated Learning with different number of clients and different amounts of data as well as the difference in performance between Tensorflow Federated since both currently are frameworks mainly for research purposes.

4.3.1 Fixed total number of data

We choose MNIST as the dataset to be tested in the framework as it is the most popular machine learning framework. In the first setup, we fixed the number of total data in the training dataset to be 3000. This is to inspect the accuracy and training time performance differences with different numbers of clients. The result are shown below:

Total number of data: 3000
Number of rounds trained: 3

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 4.1 *Evaluation Results using the Sherpa.ai framework with fixed total number of data*

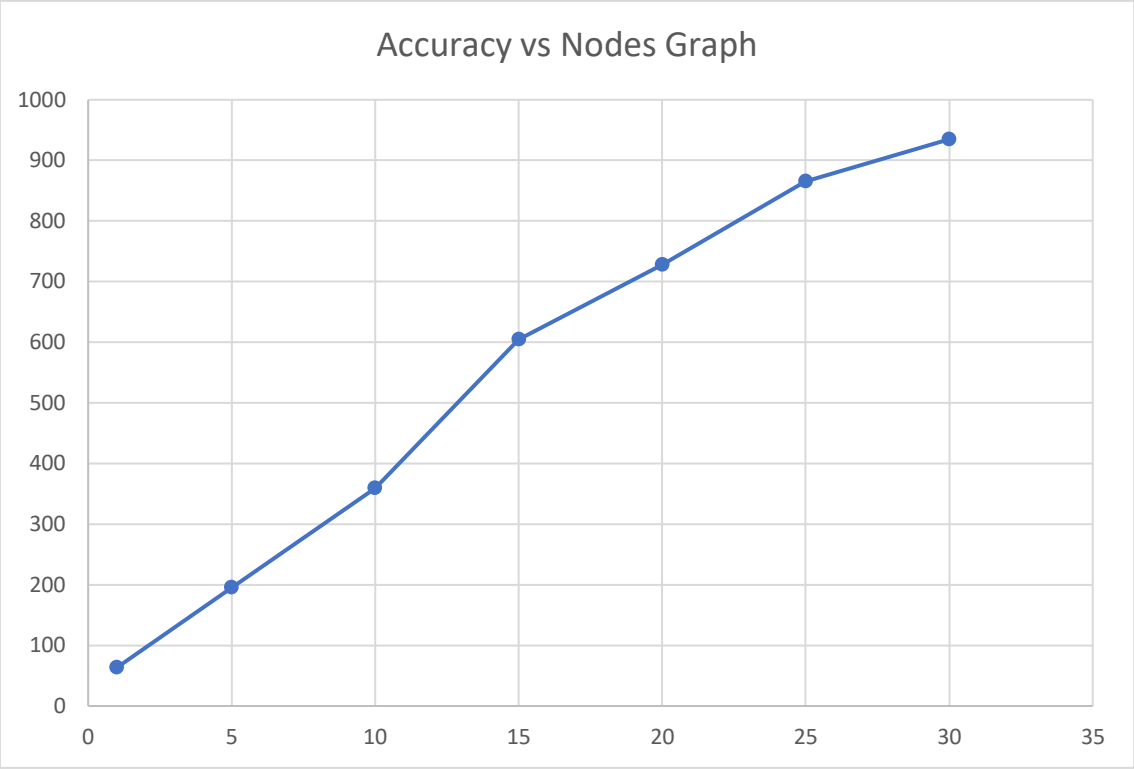

Figure 4.21 *Accuracy vs Nodes Graph for Model with fixed total number of data*

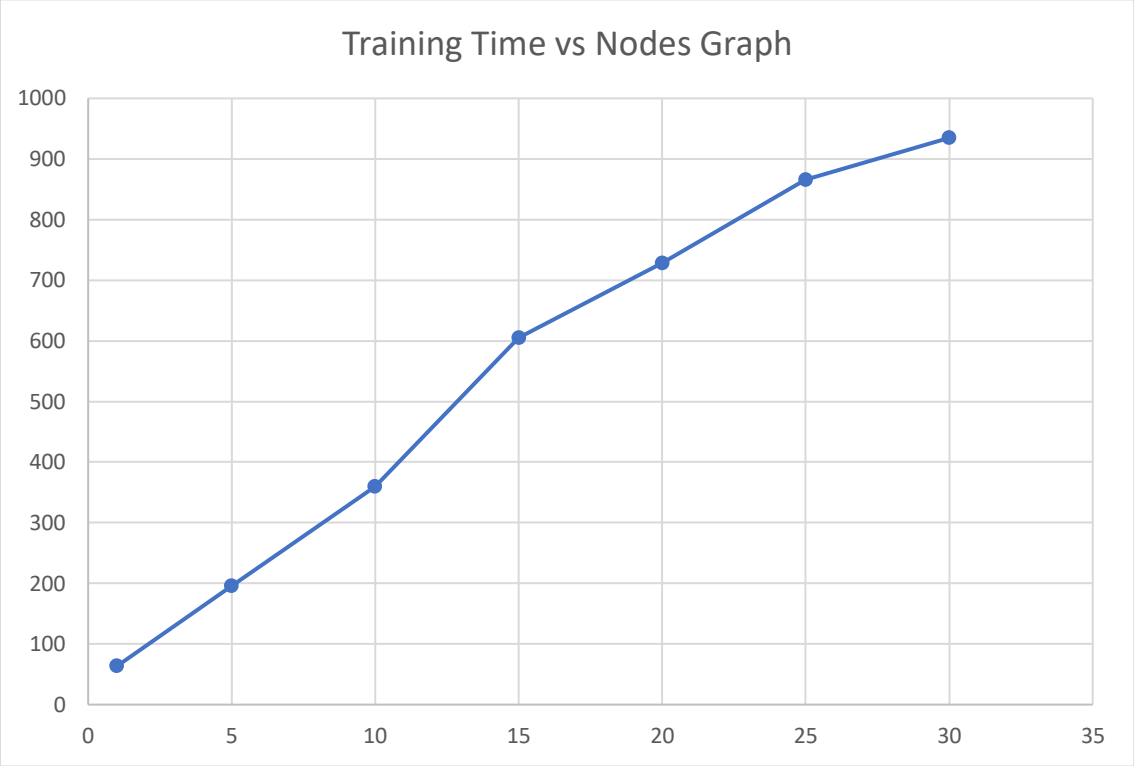

Figure 4.22 *Training Time vs Nodes Graph for Model with fixed total number of data*

We can see that although we have the same total number of data to be trained, if there is more client, i.e. the data are more scattered, the accuracy will drop significantly.

On the other hand, even with the same amount of data to be trained, since there is more client to perform training. There is a linear increase with the training time and number of clients.

4.3.2 Fixed data per client

In the second setup, we fixed the number of data per client in the training to be 300. This is to inspect the. The result are shown below:

Number of data per client: 300
Number of rounds trained: 3

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 4.2 *Evaluation Results using the Sherpa.ai framework with fixed number of data per client*

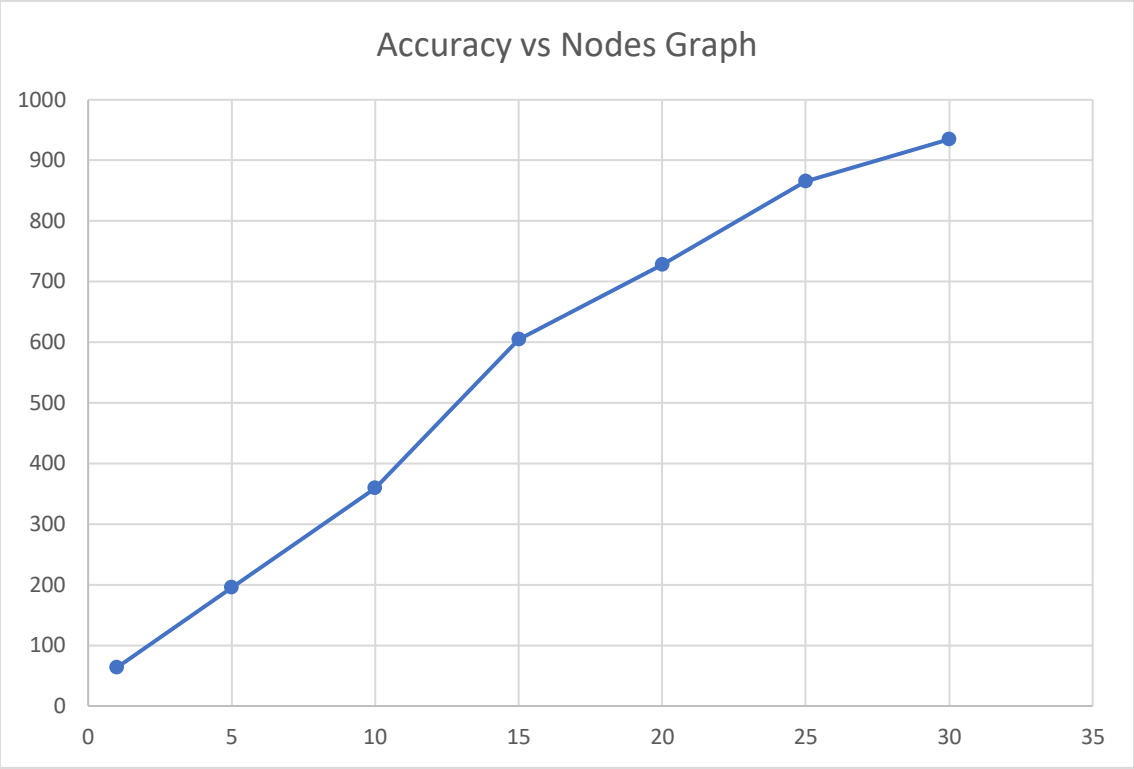

Figure 4.23 *Accuracy vs Nodes Graph for Model with fixed total number of data*

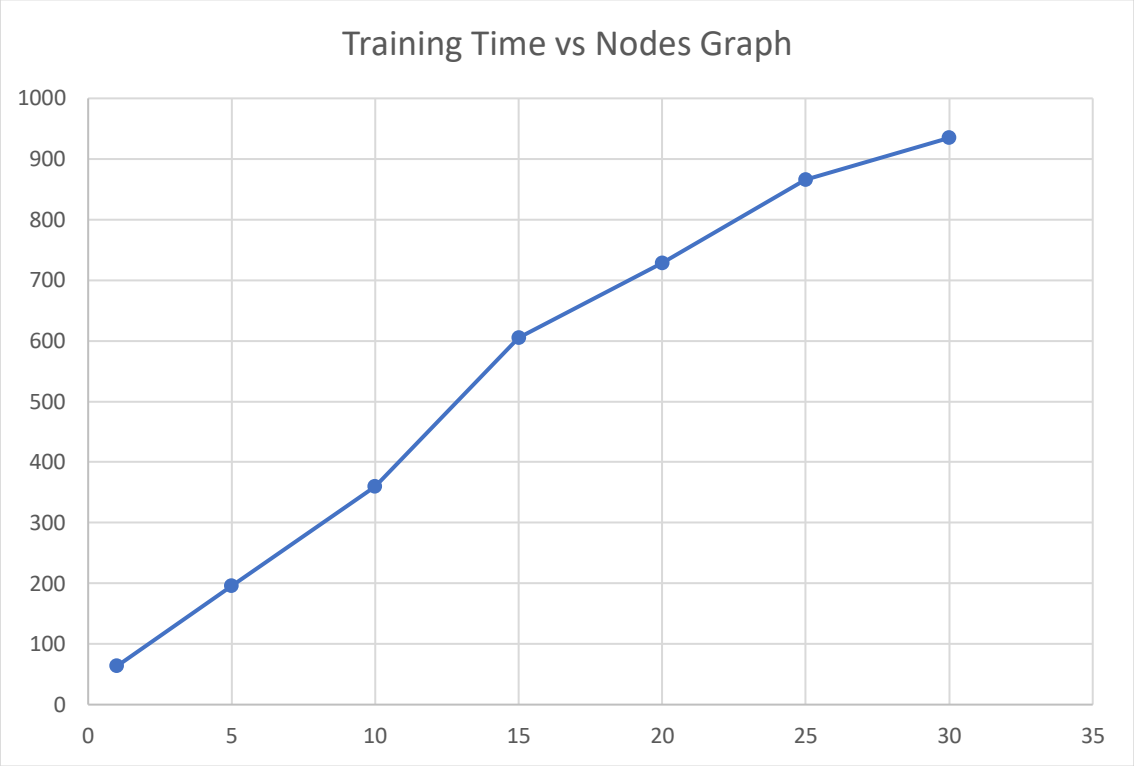

Figure 4.24 *Training time vs Nodes Graph for Model with fixed total number of data*

As expected, when there is more total number of data to be trained, the accuracy gradually increase.

There is an interesting finding which is the training time with fixed data per client and fixed total data is quiet similar, although when there is more clients, the difference is total data is much larger, as shown in Table 4.3. This indicate that the amount of data is not the main factor for training time, the amount of clients is the key factor since the more client it has, the more training rounds needs to be performed. However, this result are obtained using local simulation where all clients are doing training on one single device. In reality, when client perform training on their own device, such issue will not happen and we expect to see a uniform training time and the key factor for training time will then become the amount of data present within the client.

No. of nodes (clients)	Fixed total number of data		Fixed data per client		% Difference
	Data per client	Training Time(s)	Data per client	Training Time(s)	
1	3000	74.85	300	63.41	16.5485%
5	~600	206.66	300	195.66	5.46828%
10	~300	386.08	300	359.30	7.1856%
15	~200	461.30	300	604.89	26.9352%
20	~150	721.09	300	727.94	0.94546%
25	~120	903.46	300	865.50	4.29179%
30	~100	960.65	300	934.86	2.72117%

Table 4.3 *Training time of Sherpa.ai framework with fixed total amount of data and fixed amount of data per client*

4.4 Flower

A Federated Learning command line interface platform was developed using the flower framework in Python. The platform contain a host server and different number of clients where the clients will interact with the server during the training process. The server oversees the whole training process and select specific clients

for each round of training. Clients will connect to the server and will be selected by the server. When the client is selected, the server will send the updated model at the start of each round of training. When the global model is received by the client, it will train the global model locally using its own data. The communication between server and clients is handled by the flower framework in the background.

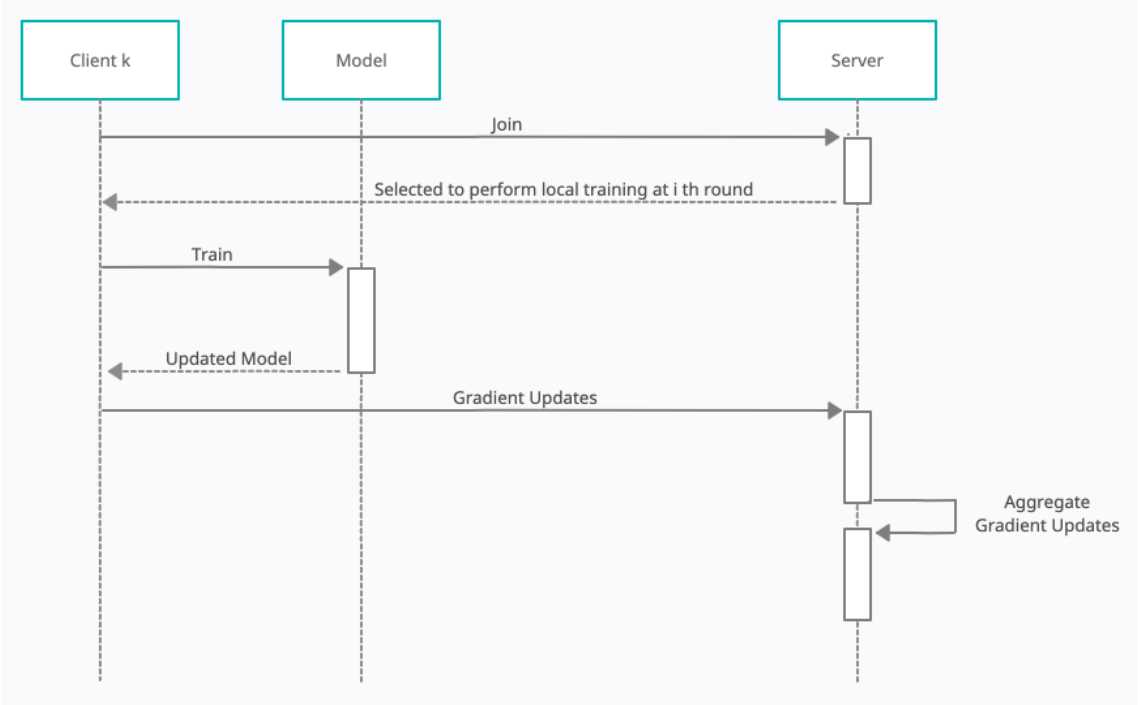

Figure 4.25 *Sequence Diagram of one round in the FL CLI platform*

4.4.1 Rationale for using Flower

Scalability, compatibility with edge devices, proven infrastructure, and usability are the four main reason the flower framework is chosen [7].

For scalability, Flower can handle real-world setups with a large number of clients. Researchers have tested with over 10,000 clients with this framework, and it makes the platform scalable for large scale project [7].

High compatibility with edge devices allow FL to be performed on different servers and devices, including mobile and PCs. Android, iOS, Raspberry Pi, Nvidia Jetson are some examples that are compatible with Flower. Different

operating systems and hardware platforms are able to access and work with heterogeneous edge devices.

For proven infrastructure, Flower provides FL infrastructure to ensure low engineering effort which allows the developers to ignore the FL infrastructure and concentrate on the ML aspect instead.

In the aspect of usability, flower easy to understand and write custom algorithms that fulfils the purpose of the federated learning platform especially it is written in python.

4.4.2 Evaluation

The server have an evaluation function that will get called after each round of training and will perform testing on the updated model. A federated averaging strategy will selected 10% of the total clients connected to the server with a minimum of 3 clients per round, and ensure there are more than 4 clients connecting the server at all times. The client will have 500 train samples and 10 test samples before the training process. CIFAR10 and MNIST dataset are chosen to test the platform locally in the Command Line Interface (CLI).

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 4.4 *Evaluation Results using the CLI platform*

The MNIST dataset and CIFAR10 both consist of 10 classes but the MNIST have only one channel while CIFAR 10 have 3 channels. With all other factors are kept constant to compare the datasets, the training done on CNN is slower to that of FCNN because CNN needs more computations in each pass. The significant difference in accuracy is because FCNN model have fewer trainable weights compared to CNN and therefore converges to the ideal weights at a faster rate.

No. of Clients	No. of Rounds	No. of min. Fit Clients	Accuracy
5	5	2	82.11%
5	10	2	85.50%
5	15	2	86.04%
5	20	2	86.47%

Table 4.5 *Evaluation Results on MNIST with constant client number*

No. of Clients	No. of Rounds	No. of min. Fit Clients	Accuracy
2	10	2	81.67%
3	10	2	84.15%
4	10	2	84.94%
5	10	2	85.50%

Table 4.6 *Evaluation Results on MNIST with constant number of rounds*

No. of Clients	No. of Rounds	No. of min. Fit Clients	Accuracy
10	2	2	81.67%
10	3	3	81.83%
10	4	4	84.25%
10	5	5	83.81%

Table 4.7 *Evaluation Results with constant rounds and different number of minimum fit clients*

4.4.3 Deployment

The platform can only runs on the CLI locally right now. The flower framework is currently in its early stage and is only capable of on-device simulations. The framework cannot be deployed yet without the evolution of the framework. Extensive testing needs to be done to prove the platform is secure enough from data leakage.

4.5 PySyft, PyGrid and Syft.js

4.5.1 PySyft

PySyft is a python library developed for secure and private Machine Learning. It separates the private data from the training of the ML model using the technology of Federated Learning, Differential Privacy, and Encrypted Computation with PyTorch and TensorFlow.

PySyft server two main purposes:

1. Dynamic Computation: It directly compute data that is private
2. Static Computation: It creates static computational graphs that can be deployed later on different computes.

4.5.2 Duet

Duet is a peer-to-peer tool in the PySyft library that provides an API to allow a Data Owner (client) to privately expose their sensitive data while a Data Scientist (server) can access or manipulate its data with zero-knowledge access control mechanism. It is a novel approach that stray away from the conventional central server management method in FL where data owners control their own data and ensures effective communication between data owner data scientists. Data owners have the right to decide when to participate in the training process and also what operations can be performed on their own data by the data scientists.

Duet is used to demonstrate training a model using PySyft without deploying in PyGrid. However, Duet only supports a single client. Therefore, it is not possible

to create multiple clients and train on their data simultaneously. To simulate multiple clients in a FL environment, training needs to be done one by one. MNIST dataset is used which contains 60000 training images and 10000 testing images. We fixed the total number of data to be 60000 training images. We first train the model with one client. Then, we simulate multiple clients by training the model in multiple sections one by one.

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

Table 4.8 *Evaluation Results with constant rounds and different number of minimum fit clients*

# Clients	# Data per client	# epochs	Accuracy	Time (s)
1	60000	10	98.00%	2156
5	12000	10	96.46%	2588
10	6000	10	94.48%	3139

Table 4.9 *Evaluation Results with constant rounds and different number of minimum fit clients*

Result in Table 4.7 and 4.8 are similar to what we seen in subchapter 4.3.2 in the Sherpa.ai framework with fixed total number of data. The accuracy drops when there are more clients, or the data is more decentralized. However, we can see the drop in accuracy is less significant in duet compared to that of Sherpa.ai.

4.5.3 PyGrid

PyGrid is another python library that provides an API for the management and deployment of PySyft at scale. It enables the extension of PySyft to perform FL on

jupyter notebook, web, mobile, and similar edge devices with different Syft worker libraries.

4.5.4 Syft.js

Syft.js is a frontend PySyft worker library in TensorFlow.js for FL which supports federated learning on the web. It provides APIs to communicate with FL PyGrid endpoints and run PySyft's Plans in a browser. It integrates with PyGrid FL API. It also supports training and inference of PySft ML models which are written in PyTorch or TensorFlow. It allows data to stay on the user's device as the core feature of FL. However, the security protocol of Federated Learning such as Secure Multi-Party Computation and secure aggregation using peer-to-peer WebRTC connections is still in progress.

4.6 Challenges

4.6.1 New Technology

One of the challenges that this project faces is that the technology concept behind which is Federated Learning is relatively new in the industry. Therefore, there will be fewer references, materials that the project can reference. There will be a very high chance that throughout the project, it will encounter many problems that have never been asked before. Moreover, some obstacles that the project might face could very likely not be solved or require complex solutions that are beyond the group members' capabilities.

4.6.2 FL Library still in early stage of development

For Tensorflow Federated, the framework is still in early-stage development. It only supports local simulations, Multi-machine simulations are yet to be developed. The purpose of the framework right now is mainly focusing on research and evaluation of FL instead of deploying FL systems at scale. Therefore, in our project, Tensorflow Federated are only used to demonstrate the performance difference between traditional ML and FL as well as how FL differs from conventional ML.

Flower has also the above issue where it doesn't support multi-machine simulations and therefore is not able to be deployed as mentioned in subchapter 4.4.3.

4.6.3 Generalization of different types of FL

The platform cannot be generalized to handle all types of ML algorithms. Different types of federated learning as mentioned in subchapter 1.2.1 required different architectures [2]. Therefore, the platform developed will only handle horizontal FL.

4.6.4 Complexity of implementing of Security Protocol

Recent researches have shown that there might be potential security breaches when training the platform using Generative Adversarial Network [2]. Hence, it will be impossible to ensure data privacy for GANs and hence cannot be implemented.

5 Future Work

5.1 Future Work

There are two main categories of work to be done in the future - development of a separate web application, and testing of security features.

5.2 Deploy web platform

As mentioned in subchapter 4.6.2, most of the library and framework are not matured and ready for deployment at scale yet. It is mostly for local simulation and currently in the project, most of them are done in local environment, with the exception of syft.js.

5.3 Testing on security

When the platform is deployed on the web, there should be testing to see whether the security of the platform is up to standard. Moreover, different security protocol should be tried on the platform and compared to see which have the highest security as well as performance.

6 Conclusion

Traditional Machine Learning model training requires all data to be centralized. Due to the strict data privacy of different industries and countries, it is hard to collect from different parties and achieve effective training of the ML model. Federated Learning is a new concept that arose in 2016 that allows decentralized ML training, enforcing data privacy. Our project aims to build a Horizontal Federated Learning platform where participants can collaboratively train a machine learning model. This project, if carried out successfully, will hope to be used by AI InnoBio, a company that aims to use FL to conduct rapid Covid-19 tests using a CMOS sensor. We have done research and evaluation on different federated learning methods in Tensorflow federated, Sherpa.ai and developed a basic client-server communication using Python Flower framework, duet and syft.js and talked about what can be done to improve in the future.

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].