The University of Hong Kong
Department of Computer Science

Federated Learning Platform for COVID-19 Detection

Kwan Pok Man 3035477173

Teammates:
Ghosh Bratin 3035437692
Khan Khondoker Araf Khan 3035477446

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

18 April 2021

Abstract

Technologies nowadays allow computer to perform lots of complicated tasks accurately and efficiently. One of the fastest growing technologies is machine learning, computers are able to act and make smart decisions like human. Machine learning requires a lot of data during the training process so that the computer can find out the hidden relationships and patterns, and generates the outputs based on its experiences. The amount of data in our world is increasing exponentially every day, which is essential for training machine learning models with great performance. However, data privacy is getting far more attention than ever before. Governments and organisations are not willing to share private data to other parties for any purposes, including machine learning. Therefore, a new way of training machine learning models without gathering data to a central server, and without 'seeing' the actual data, is needed. This leads to the development of Federated Learning, a new technique proposed by Google in 2016. The objective of this project is to understand Federated Learning and build a platform for users to train a machine learning model without centralizing the data. We have tried out several Federated Learning frameworks and tried to deploy them.

Acknowledgements

First, I would like to thank Professor S.M. Yiu for his kind support, and for giving us the opportunity to conduct this final year project.

I would also like to thank my teammates, Araf and Bratin, it is my pleasure to work with them.

Lastly, I would like to thank Miss Mable Choi for the clear guidance on writing this report.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

With the advances in computational power and the exponentially increasing amount of data, artificial intelligence (AI) technologies are becoming more popular and more useful in many areas. Today AI has better performance than humans in many tasks, for example, diagnosing diseases, predicting stock prices, or even playing video games. One may think that it is the improvements in machine learning algorithms that lead to the success of these AI applications. This is true, but the data behind these AI are of equal importance. Most machine learning models are trained with a substantial amount of data, so that they can generalize the features from the data to make accurate predictions or decisions. Without sufficient data, the performance of many AI would be greatly reduced, and they would not be useful anymore.

Traditionally, machine learning models are trained using data in a single device or a single server. If data exist in multiple local devices, all of the data must be sent to the central server for training the global model. The cost of transporting and processing with such a large amount of data would be very huge, and most of the time it is not possible to gather all the required data to train a machine learning model. Relying on a stable network connection and the limitation for real-time applications are other possible challenges for traditional centralized machine learning.

Moreover, there is an even bigger challenge for traditional centralized machine learning. Governments and the general public are paying more attention to data privacy than ever before. Personal information is required by many websites and mobile apps to verify users' identity and to provide better services for users. People are worried that their information is used by unauthorized companies or governments for unknown purposes. Therefore, many countries and companies have laws and regulations to protect personal data. This isolates data in different parties, making it very difficult to gather data for training a machine learning model.

1.2 Introduction to Federated Learning

To overcome the challenges mentioned above, an algorithm that allow different parties to train a machine learning model without sharing the actual data is needed. Google proposed a concept called Federated Learning (FL) in 2016. Federated Learning (FL) is a technique to train machine learning models on multiple local devices without exchanging data between them. In FL, local devices download the current global model from a central server, and train it with local data. The updated local model is then sent back to the central server, where it aggregates with the global model, and produces an improved global model. This improved global model is then sent back to all local devices when they are available. More details about FL will be discussed later in this report.

1.2.1 Benefits of Federated Learning

First of all, federated learning provides a way for users to train machine learning models collaboratively without exposing their data to others. This allows organisations such as hospitals or banks to perform machine learning without giving out sensitive data.

Federated learning also reduces the time and cost for data transmission, because data never leaves the devices, only model weights that are much smaller in size compared to data are transmitted to the server through the internet. This also allows the model to be trained with the local data without internet.

Federated learning only requires a minimal hardware, so that mobile phones and small hardware systems are able to participate in federated learning in real-time [3]. One example is Google's Gboard on Android systems, which will be discussed later.

1.2.2 Challenges of Federated Learning

Although devices can train their model locally, internet is essential for improving the efficiency of the whole federated learning system, and get a more accurate global model. Therefore, a secure and stable network is required in order to reduce the total number of rounds of communications to be more efficient.

Data in different parties might be very different. For example, one device has a lot of dog images, whereas another device has a lot of cat images. This leads to a very unbalanced dataset. Furthermore, not all devices are active at anytime, so the participation rate might be very low. A good federated learning system should be able to handle both of the above situations.

According to research, federated learning cannot guarantee 100% privacy because the sharing of model weights might have a chance of revealing sensitive information [2].

1.2.3 Examples of Federated Learning

Federated Learning is still a relatively new concept in the industry. Google's keyboard on Android is one of the products that has already applied federated learning.

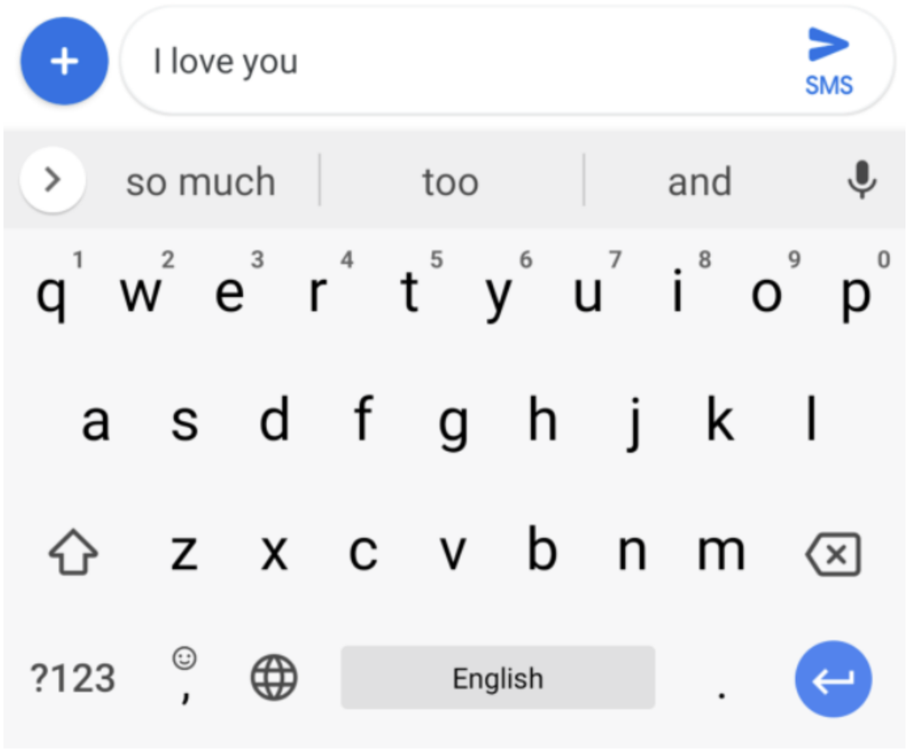


Figure 1.1: Word Prediction in Gboard [1]

Google would like to make real time and accurate word predictions based on previous words inputs. In order to do so, a large amount of data from users around the world is required. However, users do not want Google to know what they typed, so Google chooses to use their own technique, federated learning, to train a model without getting the context from the Android devices. Other than Gboard, federated learning is also being applied to Apple's Siri [4], and it can be applied to a lot of different areas such as FinTech, insurance, or healthcare in the future.

1.3 Objectives

In December 2019, a new kind of coronavirus (SARS-CoV-2) was identified in China. Since then, the coronavirus disease 2019 (COVID-19) has been ravaging the world, causing 140 million confirmed cases, and 3 million deaths as of 17 April 2021. To slow the spread of COVID-19, the best way is to identify the infected people and isolate them. Currently, a few hours are required to get the test result. A biotechnology company in Hong Kong, AI InnoBio Limited, wants to adopt an industry- leading CMOS sensor technology with a machine learning model, to perform a fast, accurate, and low-cost COVID-19 saliva test. This technology was tested in Israeli with hundreds of patients, and it achieved an accuracy of 95% of identifying the presence of the virus in just a few seconds. [5]

The machine learning algorithm and the data are not shared with the company, due to regulations and privacy. In order to use the technology in Hong Kong, new data need to be collected to train a new machine learning model. However, due to the low number of confirmed cases in Hong Kong, there is not enough data to train an accurate model. Also, data from other countries are difficult to collect due to their privacy laws. Therefore, a technique to train machine learning models without gathering data to a single place is required, so that different countries can train their models locally without sharing their data, and the models are then combined together to form the final model.

This project aims to explore FL and tries to apply it to COVID-19 detection. In the first part of the project, different types of FL techniques will be studied. Then simple platforms for FL will be implemented using several open-source FL frameworks. The accuracy, efficiency, and privacy of the platforms will be evaluated.

1.4 Outline of Report

This report consists of 5 chapters. The first chapter introduces the background of the project, such as the limitations of traditional centralized machine learning, and the basic idea of federated learning. It also states the objectives of this project.

In the second chapter, the methodology of the project is discussed. Three types of federated learning techniques are introduced, including horizontal FL, vertical FL and federated transfer learning. A few security protocols will be mentioned briefly, followed by the benefits and challenges of FL.

The third chapter shows our experiments and results using different FL frameworks, including Tensorflow Federated, Sherpa AI, Flower, and PySyft.

The forth chapter discusses the challenges of federated learning and the obstacles we encountered during the project.

The last chapter summarizes the report and this project.

2 Methodology

2.1 Introduction

This chapter explains the techniques that will be used in this project in more detail. Section 2.2 introduces three different types of FL. Section 2.3 introduces a few security protocols. Section 2.4 and 2.5 discusses the benefits and challenges of FL.

2.2 Types of Federated Learning

Federated learning is classified into three types by Q. Yang in [2], namely horizontal federated learning, vertical federated learning, and federated transfer learning. The main difference between these three types of FL is the data distribution among different local devices. More details will be explained below.

2.2.1 Horizontal Federated Learning

When local devices have different data samples with the same feature space, horizontal federated learning is used. Figure 2.1 shows the above situation. The vertical axis denotes the samples and the horizontal axis denotes the feature space. The area inside the red dotted line is the data that can be used in horizontal FL. For example, different countries have different positive and negative cases for COVID-19, but the types of patient information are the same (e.g. age, gender, saliva sample). In horizontal FL, local devices download the global model from the central server, and then trains the model locally with local data. Encrypted results are sent back to the central server, where results are aggregated securely. The aggregated result is then sent back to local devices to update the local models. [2]

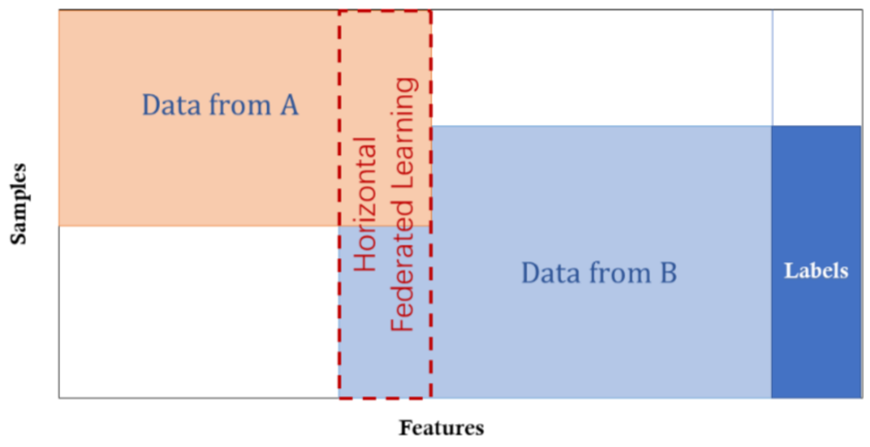


Figure 2.1: Horizontal Federated Learning [2]

2.2.2 Vertical Federated Learning

When local devices have the same data samples, but with different feature spaces, vertical federated learning is used. In Figure 2.2, it is clear that vertical FL is the opposite of horizontal FL. The dotted line surrounds overlapping samples of data from A and data from B only, instead of the overlapping features in Figure 2.1. For example, a hospital and a government may have information of the same person, but their types of information are different. A hospital could have a person's health record, while the government could have the person's travel record. In vertical FL, the two parties A and B find out their common data sample by using encryption-based techniques. Then a third party, the collaborator C, sends public keys to both A and B, and both A and B trains their local model with their data. The encrypted results are exchanged, and an additional mask is added by both A and B, and the results are sent to C. C decrypts the results and sends the results back to A and B, where the results are unmasked and used to update their model accordingly. [2]

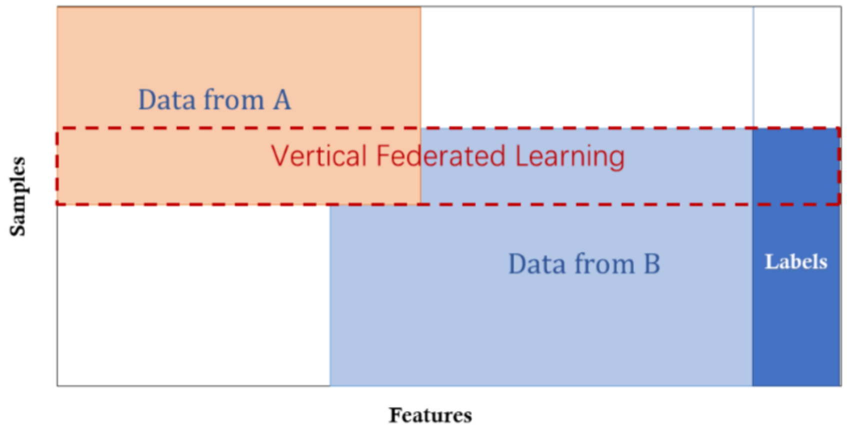


Figure 2.2: Vertical Federated Learning [2]

2.2.3 Federated Transfer Learning

When local devices have a small amount of overlapping data samples, and with different feature spaces, federated transfer learning is used. In Figure 2.3, it is shown that samples extend their feature spaces for training. This is because the overlapping area of the data is too small, to successfully train a machine learning model, new features must be learnt from other data. Federated transfer learning works similar to vertical FL, but the intermediate results exchanged between parties A and B are different. This is not the main focus of the project, so details will not be explained here. [2]

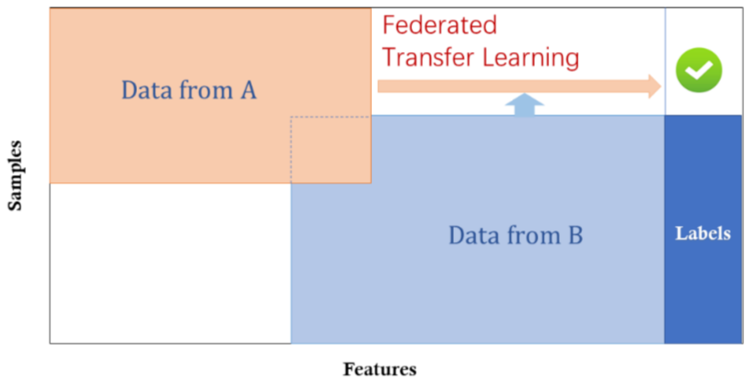


Figure 2.3: Federated Transfer Learning [2]

2.3 Security Protocols

Privacy is the main reason for developing federated learning, so a way to prevent data leakage to the server is very important. Security Protocols are used to guarantee that federated learning is safe and does not expose data to the server. In a horizontal federated learning system, it is assumed that the participants are honest and the server is honest but curious [2]. This means that the server handles all security measures and participants trusts the server. In this section, three types of security protocols will be briefly introduced, including Secure Multi-party Computation, Differential Privacy, and Homomorphic Encryption.

2.3.1 Secure Multi-party Computation

Secure Multi-party Computation distributes a computation to multiple parties, so that all parties does not have access to any other parties' data. A simple example is averaging the salary of 3 people. Each of them divides their salary into three meaningless parts, and then each person receives one part. Then the three people can add up the pieces of salary to get an average salary without the need of knowing others' salary. This ensures zero-knowledge in an FL system but it involves complex computations [2].

2.3.2 Differential Privacy

In differential privacy, noise is added to data so that outsiders cannot distinguish whether a specific individual's information is included in the computation. This can protect the privacy of the users, but the noise might modify the data and it might affect the accuracy of the machine learning model.

2.3.3 Homomorphic Encryption

Homomorphic Encryption protects the user privacy by using encryption algorithms in the machine learning process. Since the data and model are not transmitted, the possibility of data leakage is very low. However, the accuracy might also be affected when polynomial approximations are made when evaluating non-linear functions in machine learning [2].

2.4 Implementation of Horizontal Federated Learning

In this project, we are focusing on implementing a platform for horizontal FL only. There are two types of participants in the FL system, the local devices and the cloud server. According to [2] there are 4 main steps in the training process. In the first step, local devices train local models with their own data. Local devices then send the encrypted gradients to the central server. In the second step, the central server securely aggregates the received gradients without learning any information about any participants. In the third step, the central server sends back the aggregated results to local devices. In the last step, local devices decrypt the results and use them to update their local models. These steps will be repeated until the loss function converges.

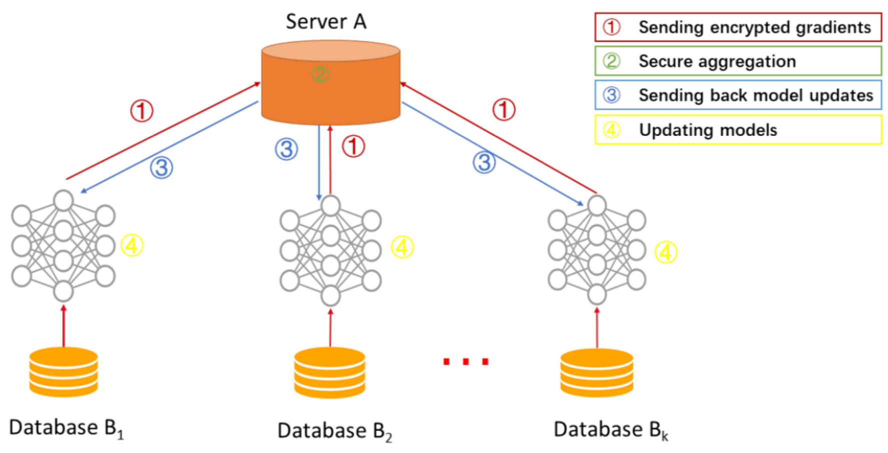


Figure 2.4: Horizontal FL Architecture [2]

2.5 Evaluation

The FL model trained with the platform will be compared to a model trained with the traditional centralized machine learning technique. Multiple open-source frameworks are used in this project. Experiments using different number of clients, different number of data per clients, different types of neural networks, and different datasets are conducted. The accuracy and efficiency of these platforms will be evaluated. Lastly, by ensuring the actual data does not leave local devices, the platform can protect data better than the traditional centralized machine learning technique. However, more investigations are required to find out whether the platform can protect the data from attacks.

2.6 Summary

Federated learning is the main focus of this project. The three types of federated learning algorithms are used in different situations according to the distribution of data. We have discussed about the basics of some security protocols and details of the implementation of horizontal federated learning.

3 Experiments and Results

3.1 Overview

In this chapter, the experiments and results are presented. This chapter is separated according to the framework used, including Tensorflow Federated, Sherpa AI, Flower and PySyft.

3.2 Tensorflow Federated

Tensorflow Federated an open-source framework for decentralized machine learning developed by Google. Three datasets were used with different models. For the Cats-vs-Dogs and CIFAR10 datasets, we used a convolutional neural network. For the Fashion MNIST dataset, we used a simple fully connected neural network.

Three different methods are used for each dataset. The first method is the traditional centralized machine learning method, using the basic Tensorflow operations. This will be used as a baseline for comparison. The second method is the Tensorflow Federated Learning API. Keras models are created and are wrapped into a tff.learning.Model, then the tff.learning API handles all the Federated Learning logics where the API is treated mostly as a black box. The third method is to build a Federated Learning algorithm using Tensorflow Federated Core. This is the same as using the tff.learning API, but we implement the functions ourselves, providing more freedom for customizations. There are four basic components in Tensorflow Federated, they are server-to-client broadcast, client update, client-to-server upload, and server update.

Below are the experiment results of the three methods and datasets.

3.2.1 Fashion MNIST

Fashion MNIST is a dataset consisting of 60000 training images and 10000 testing images. The images are in black and white. Below are some examples of images in the dataset.

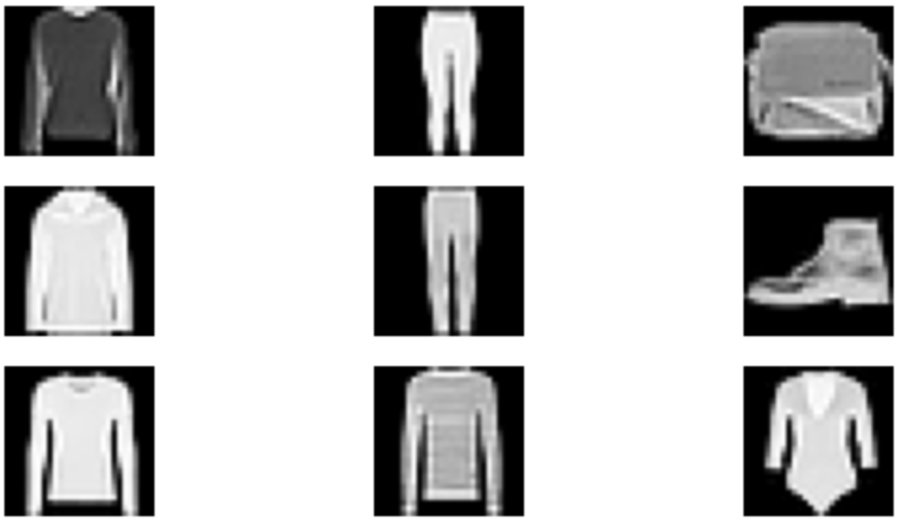
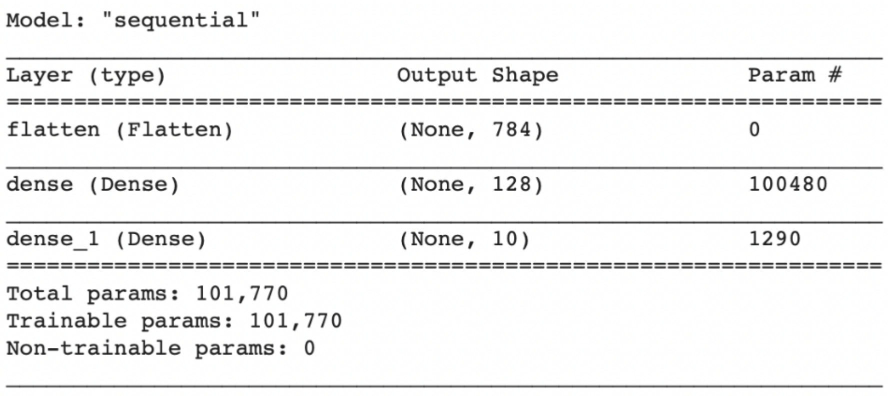


Figure 3.1: Sample images from Fashion MNIST

The images are of 28 x 28 pixels. They are flattened into a 1-dimensional tensors, and they are the input of the fully connected neural network (FCNN) below. This model is used in all three methods for the Fashion MNIST dataset.



```
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 3.2: Neural network used for Fashion MNIST

First, the traditional centralized machine learning technique is used to train the network. Only 10000 images in the training set is used. After 10 epochs, the validation accuracy reaches approximately 84%.

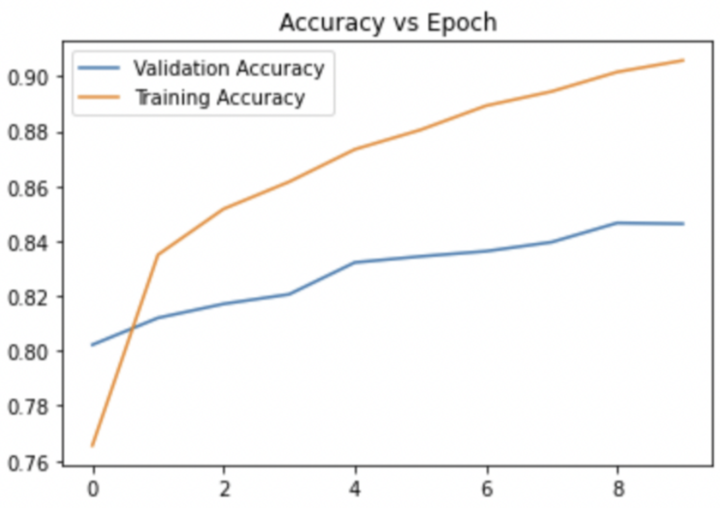


Figure 3.3: Validation accuracy of neural network trained with traditional machine learning technique on Fashion MNIST

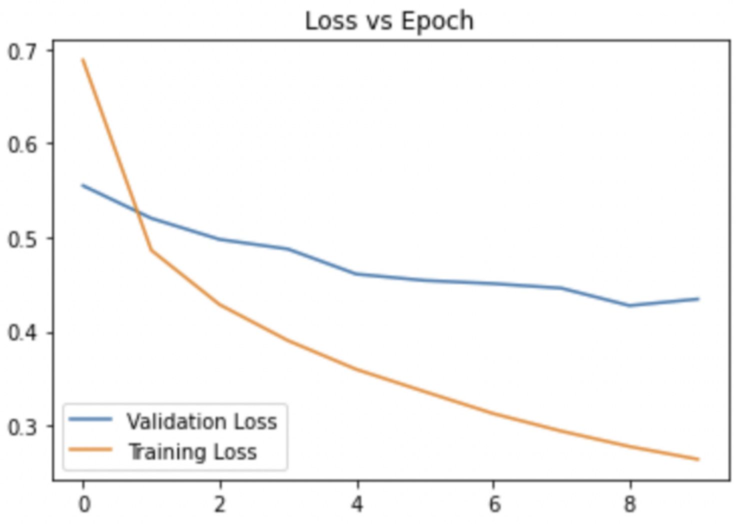


Figure 3.4: Validation loss of neural network trained with traditional machine learning technique on Fashion MNIST

Then we trained the model with the Tensorflow Federated Learning API. 100 clients are created by separating the training set randomly, allowing duplicated images in different clients. Each client has 1000 images. Below is the class distribution in the first 10 clients.

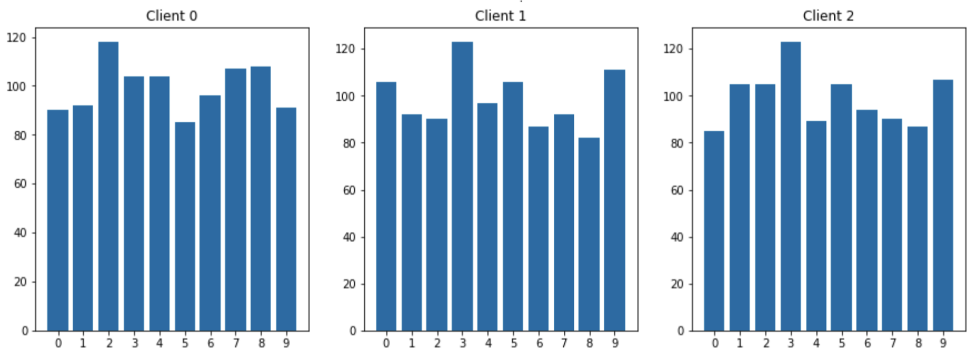


Figure 3.5: Class distribution of Fashion MNIST

10 clients are chosen in the training process, so that the total number of images used in the training is 10000 per round. Below is the validation accuracy and loss we get.

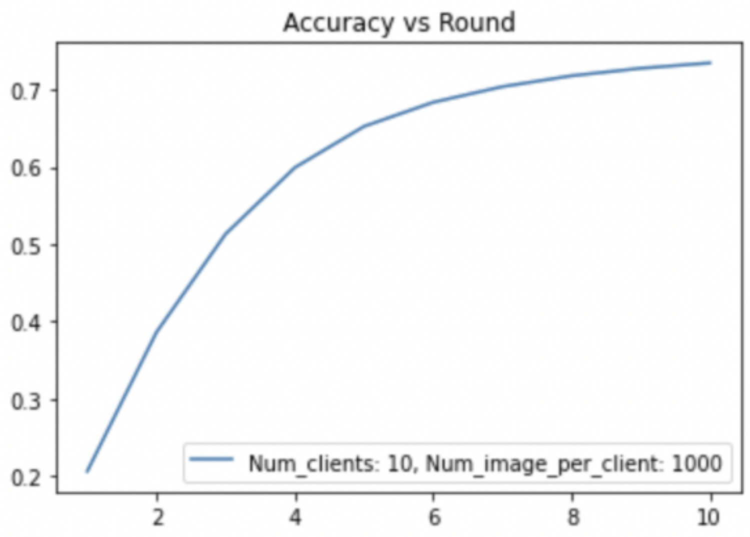


Figure 3.6: Validation accuracy of neural network trained with Tensorflow Federated Learning API on Fashion MNIST

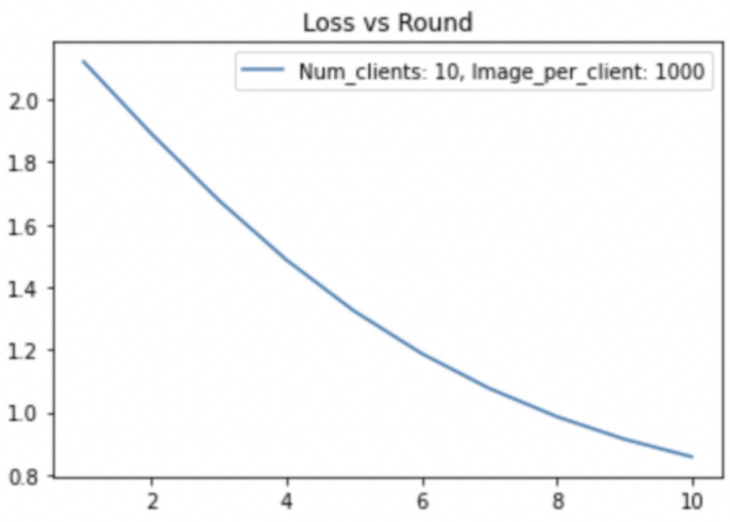


Figure 3.7: Validation loss of neural network trained with Tensorflow Federated Learning API on Fashion MNIST

The above results show that the federated learning algorithm is able to train neural network without centralizing data from multiple clients.

However, the accuracy is only 74% after 10 rounds, which is much lower than the 84% reached by the traditional machine learning technique.

Next, we implemented a Federated Averaging algorithm using Tensorflow Federated Core, and we trained the neural network with it. Below is the validation accuracy and loss we get, they are very similar to what we get from using Tensorflow Federated Learning API.

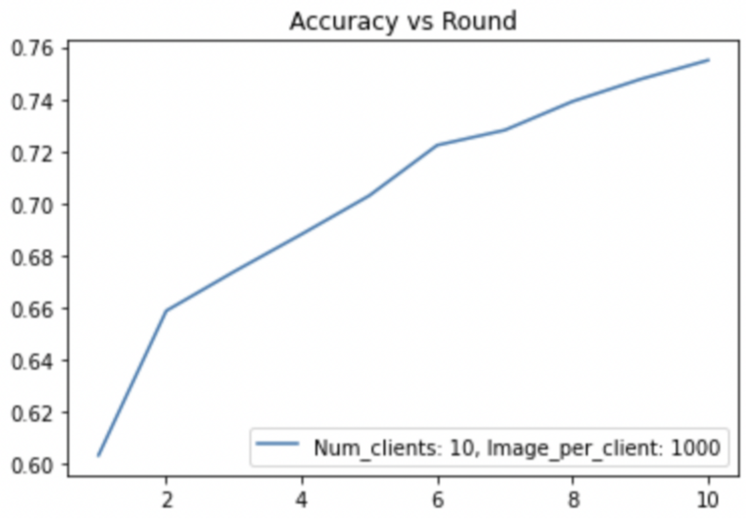


Figure 3.8: Validation Accuracy of neural network trained with Tensorflow Federated Core on Fashion MNIST

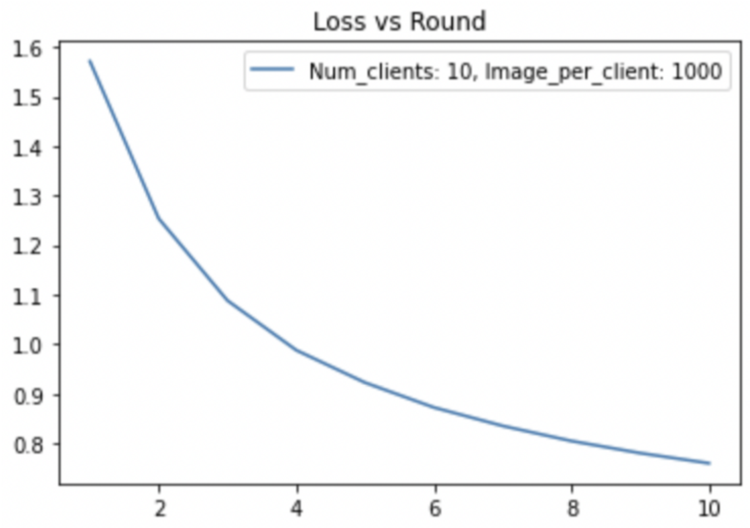


Figure 3.9: Validation loss of neural network trained with Tensorflow Federated Core on Fashion MNIST

3.2.2 CIFAR10

CIFAR10 is a dataset consisting of 50000 training images and 10000 testing images. The images have three channels. Below are some examples of images in the dataset.

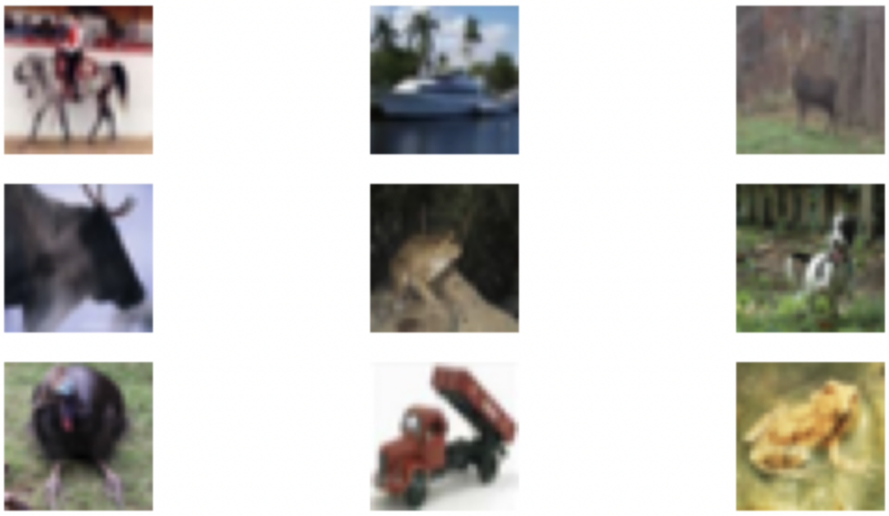
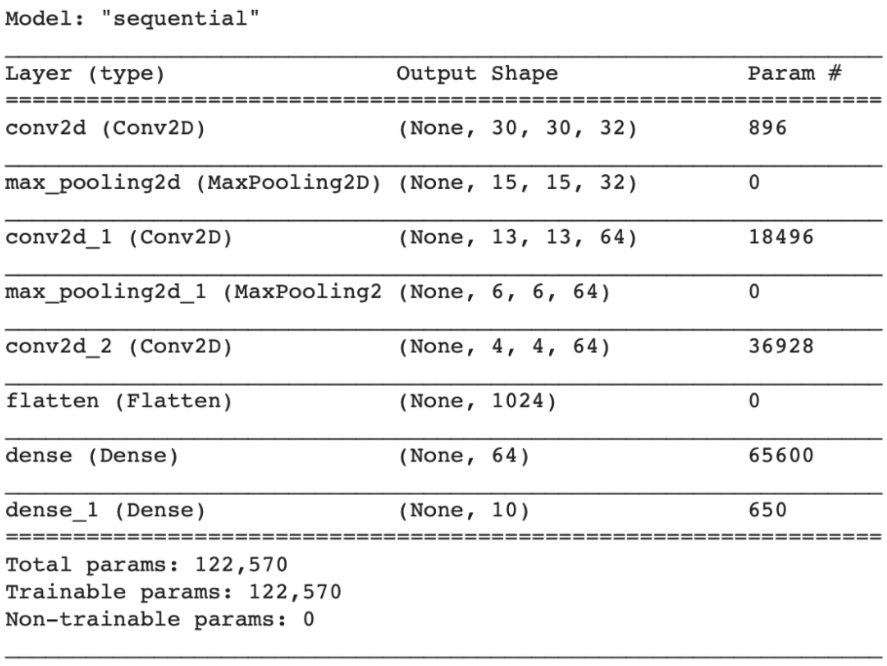


Figure 3.10: Sample images from CIFAR10

This time a convolutional neural network is used to classify the images.



```
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 3.11: Neural network used for CIFAR10

The same three methods are used to train the neural network, below are the class distributions and the results.

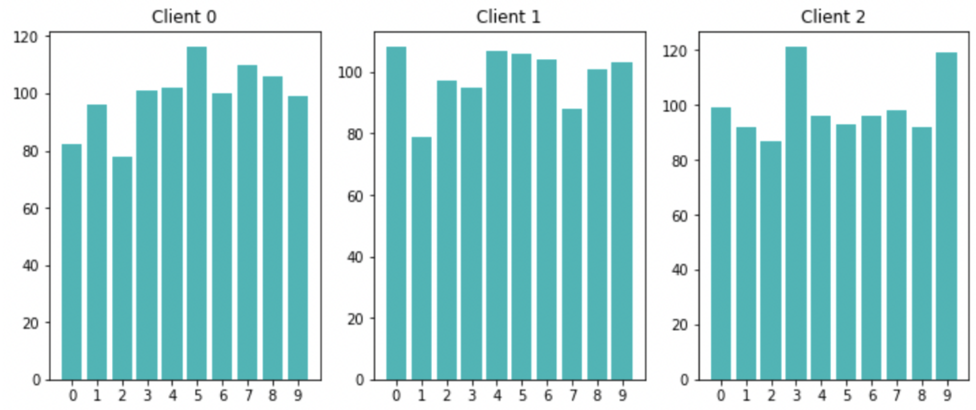


Figure 3.12: Class distribution of CIFAR10

Tradional Centralized Machine Learning:

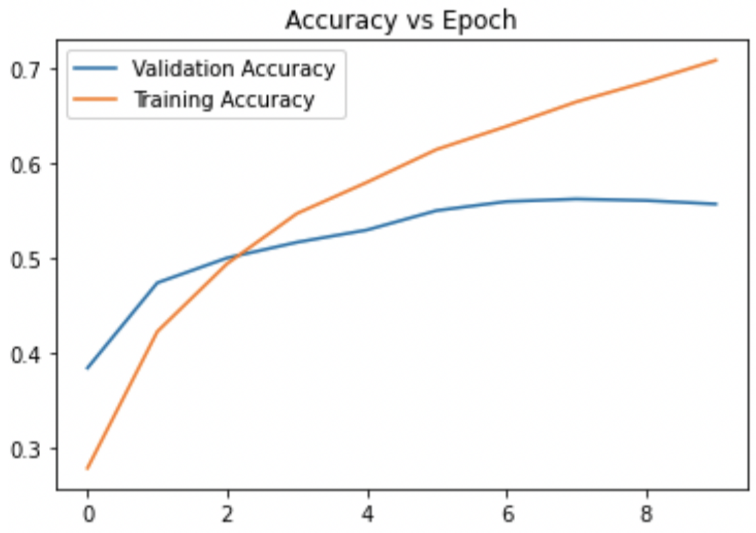


Figure 3.13: Validation accuracy of neural network trained with traditional machine learning technique on CIFAR10

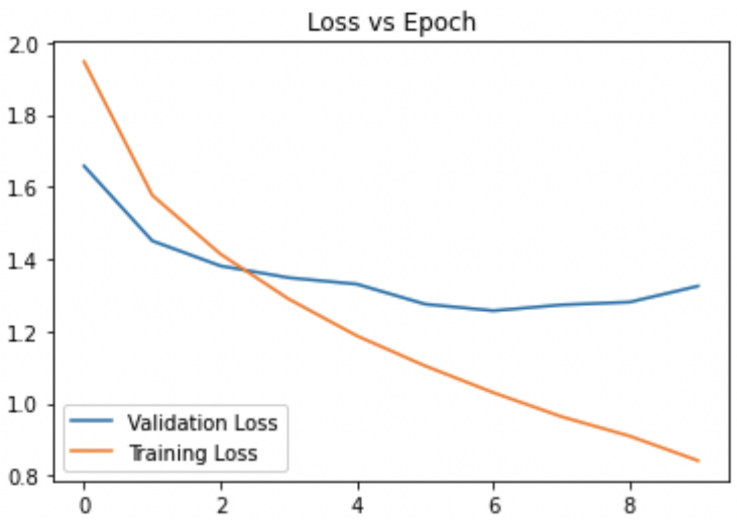


Figure 3.14: Validation loss of neural network trained with traditional machine learning technique on CIFAR10

Because of the colored and more complex images in CIFAR10, it is a more difficult problem compared to Fashion MNIST. Therefore, the validation accuracy after 10 epochs for classifying IFAR10 is much lower than that for classifying Fashion MNIST.

Tensorflow Federated Learning API:

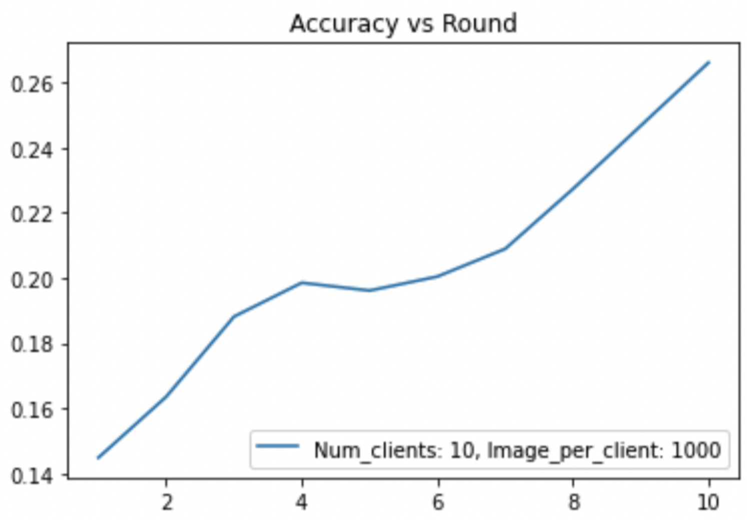


Figure 3.15: Validation accuracy of neural network trained with Tensorflow Federated Learning API on CIFAR10

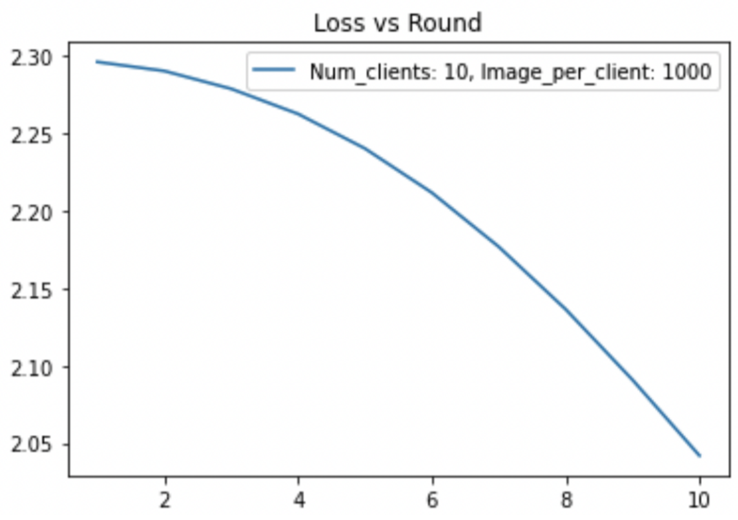


Figure 3.16: Validation loss of neural network trained with Tensorflow Federated Learning API on CIFAR10

Tensorflow Federated Core:

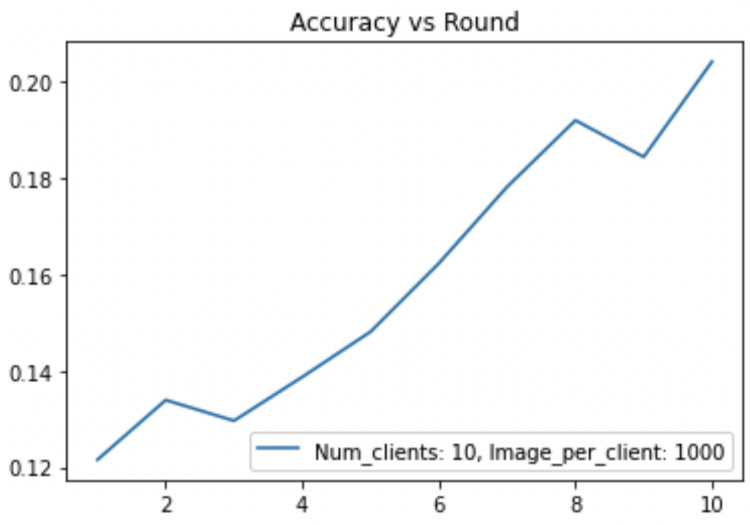


Figure 3.17: Validation accuracy of neural network trained with Tensorflow Federated Core on CIFAR10

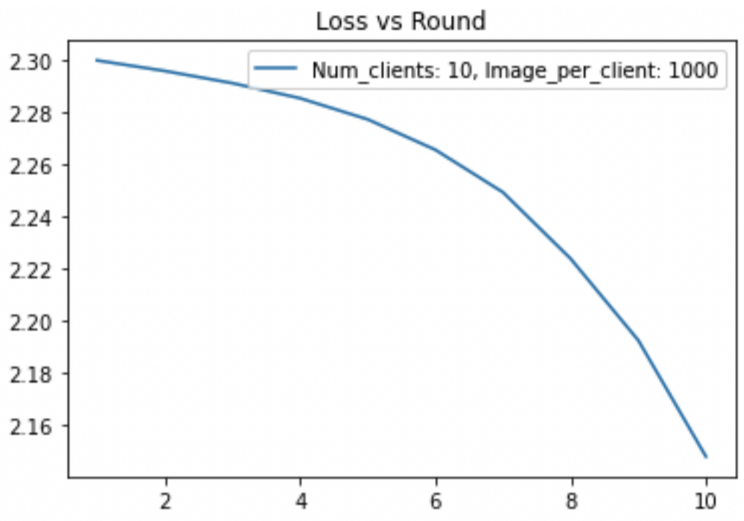


Figure 3.18: Validation loss of neural network trained with Tensorflow Federated Core on CIFAR10

From the results above, it is clear that the accuracies reached by bothe federated learning algorithms are much lower than the accuracy reached by tradition centralized machine learning for CIFAR10. One possible explanation for this is that CNN requires more rounds to converge in federated learning for more complex datasets.

3.2.3 Cats-vs-Dogs

Cats-vs-Dogs is a dataset consisting of 25000 images with three channels. Below are some examples of images in the dataset.

Figure 3.19: Sample images from Cats-vs-Dogs

This same CNN for CIFAR10 is used to classify Cats-vs-Dogs here. The same three methods are used to train the neural network, below are the class distributions and the results.

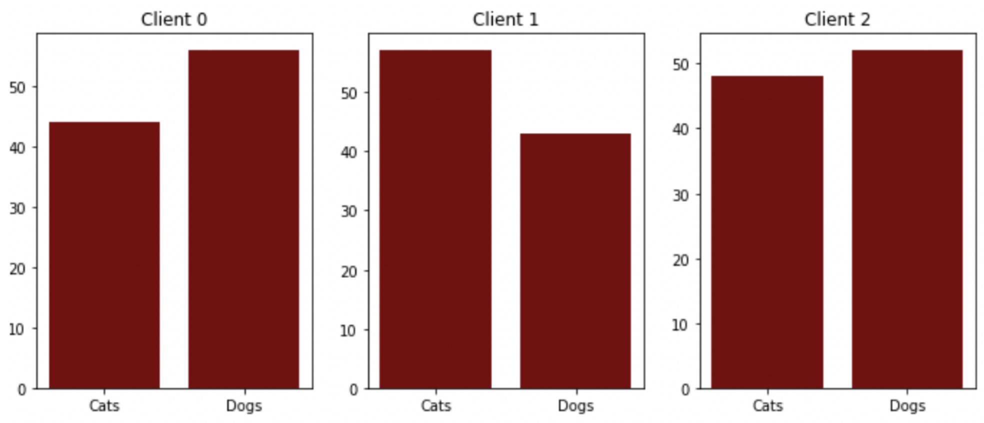


Figure 3.20: Class distribution of Cats-vs-Dogs

Tradional Centralized Machine Learning:

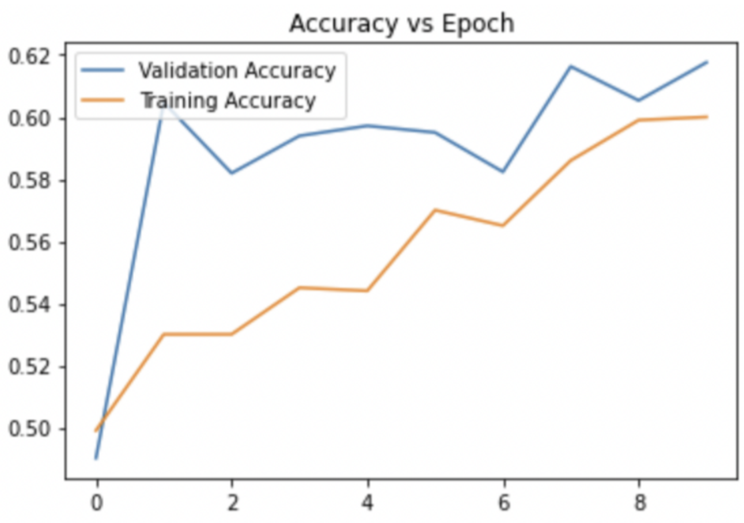


Figure 3.21: Validation accuracy of neural network trained with traditional machine learning technique on Cats-vs-Dogs

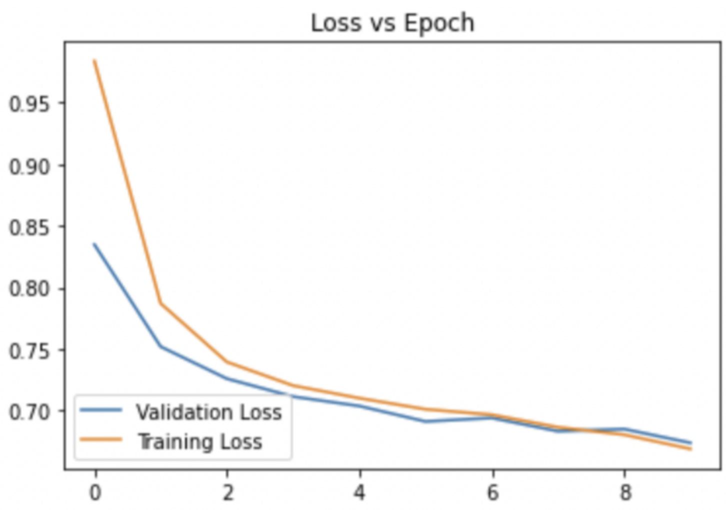


Figure 3.22: Validation loss of neural network trained with traditional machine learning technique on Cats-vs-Dogs

The accuracy reached by the neural network after 10 epoch is about 62%, which is lower than that in Fashion MNIST, but higher than that in CIFAR10. This is probably due to the nature of the dataset, as Cats-vs-Dogs is a dataset of colored images, but there are only 2 classes for classification.

Tensorflow Federated Learning API:

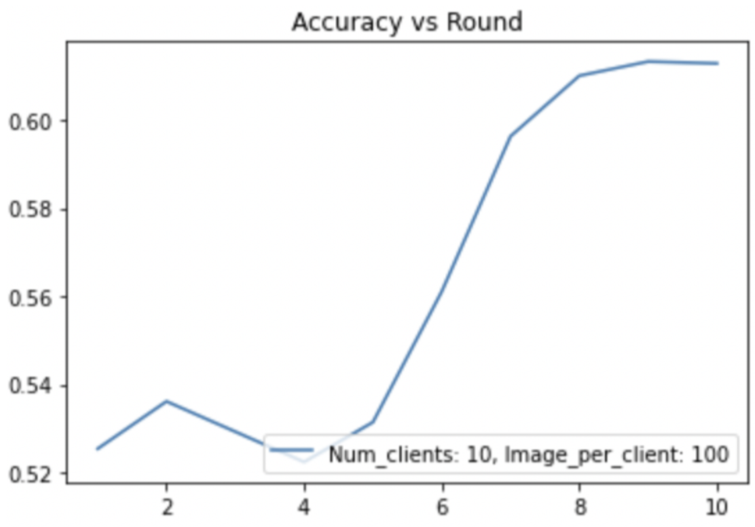


Figure 3.23: Validation accuracy of neural network trained with Tensorflow Federated Learning API on Cats-vs-Dogs

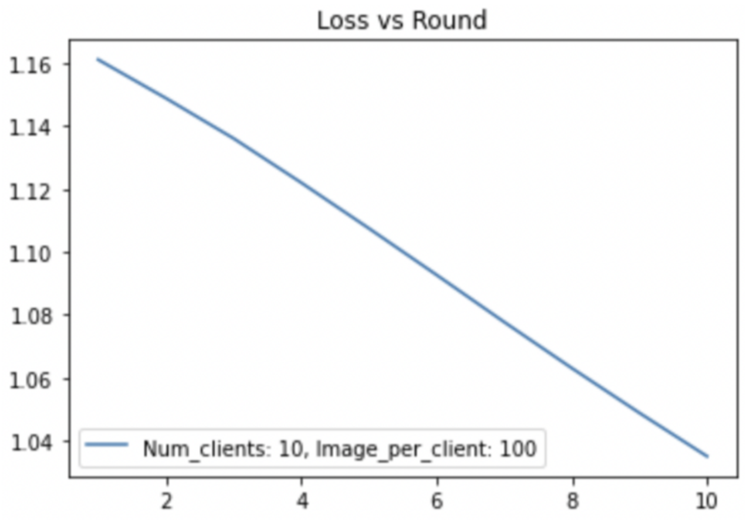


Figure 3.24: Validation loss of neural network trained with Tensorflow Federated Learning API on Cats-vs-Dogs

Tensorflow Federated Core:

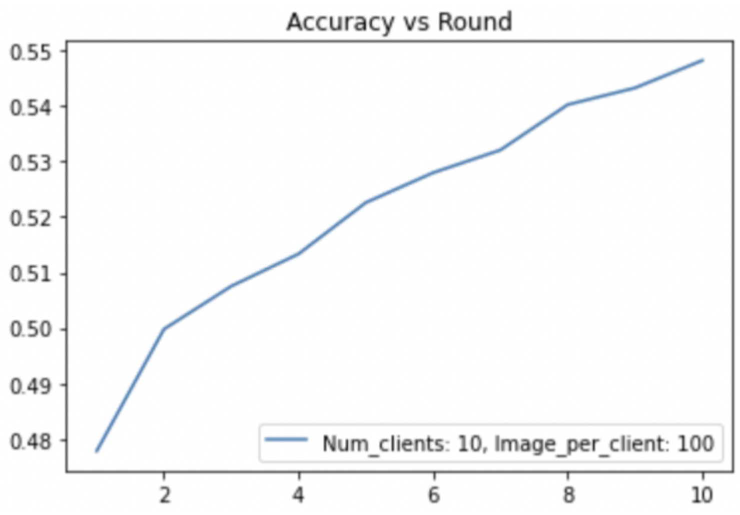


Figure 3.25: Validation accuracy of neural network trained with Tensorflow Federated Core on Cats-vs-Dogs

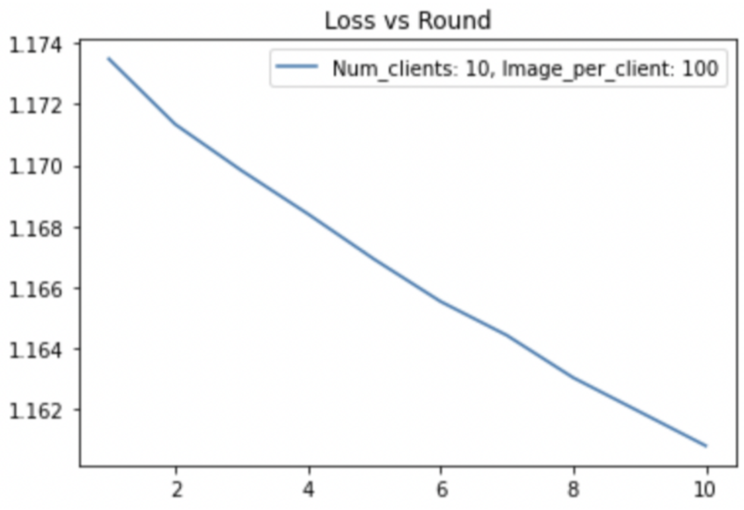


Figure 3.26: Validation loss of neural network trained with Tensorflow Federated Core on Cats-vs-Dogs

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.

3.2.4 Further experiments on Fashion MNIST

Other than changing the dataset and model structure, we would like to explore more on the effect of using different number of clients. In previous experiments, the same 10 clients are used for each round. This time we choose 10 random clients every round from the 100 clients created. Below are the results.

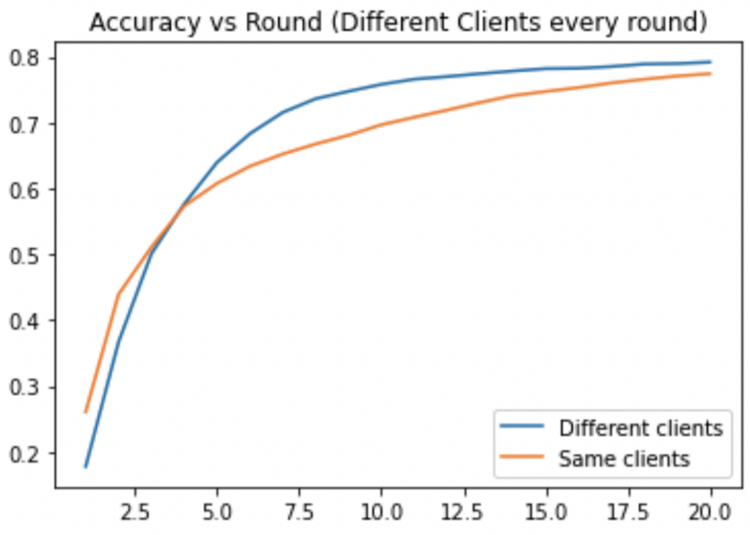


Figure 3.27: Validation accuracy of neural network trained with the same/different clients for each round

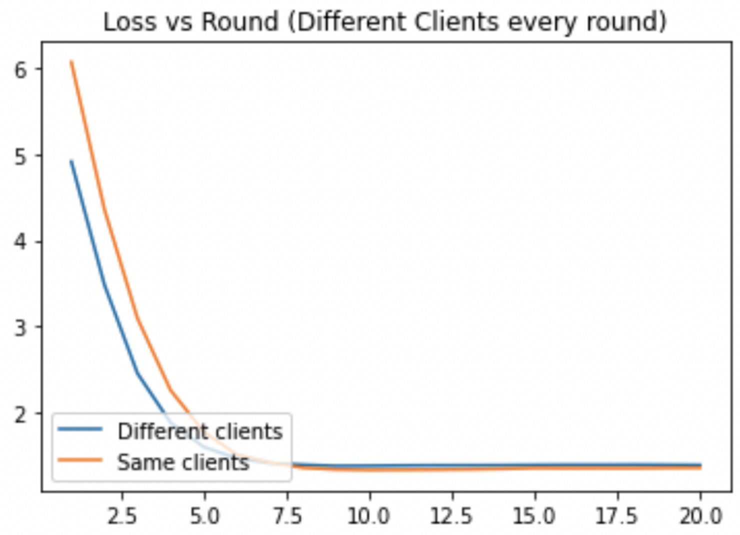


Figure 3.28: Validation loss of neural network trained with the same/different clients for each round

The accuracy of the network increases faster when it is trained with different clients, and it reaches a slightly higher accuracy after 20 rounds. This is the same as the expected result because the neural network has access to more data with different clients in each round, so it can learn faster and be better in generalization.

Next, we performed a grid search to explore the performance of the algorithm with different number of clients and different number of images per client. Results are shown below.

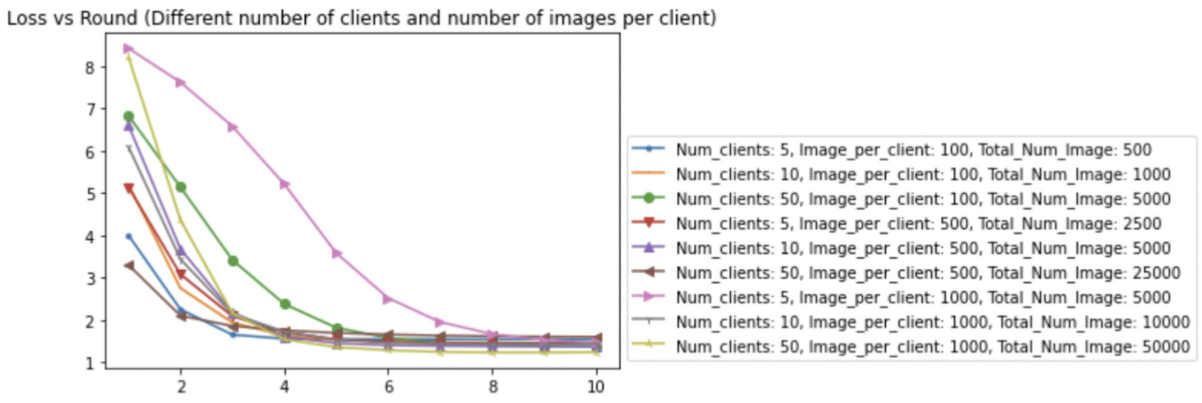


Figure 3.29: Validation loss of neural network trained with the different number of clients and number of images per clients

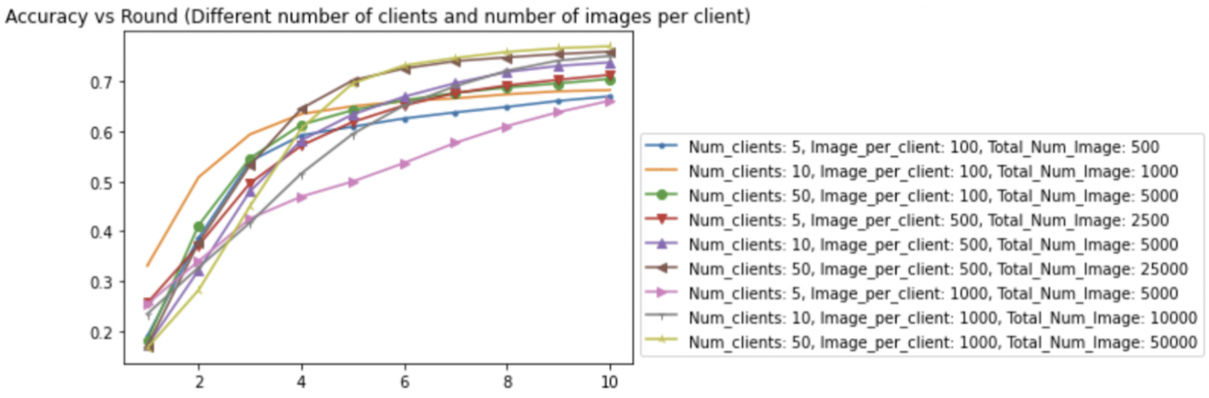


Figure 3.30: Validation accuracy of neural network trained with the different number of clients and number of images per clients

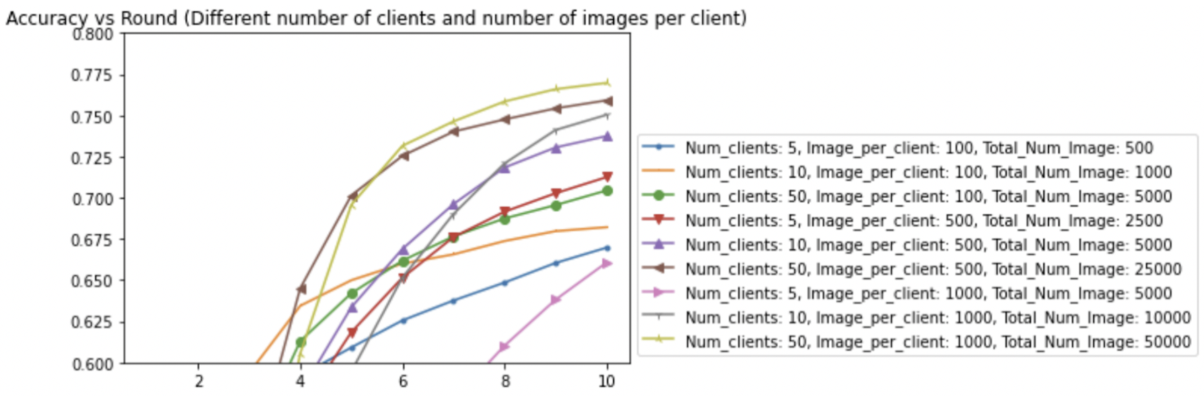


Figure 3.31: Zoom in to Figure 3.30

From Figure 3.31 we can see that in general, the accuracy increases with the total number of images used in the training. As a result, we would like to fix the total number of images used in the training and explore the effect of number of clients on the accuracy of the model. From Figure 3.32, it can be seen that the accuracy increases with the number of images per client, and when there are less clients.

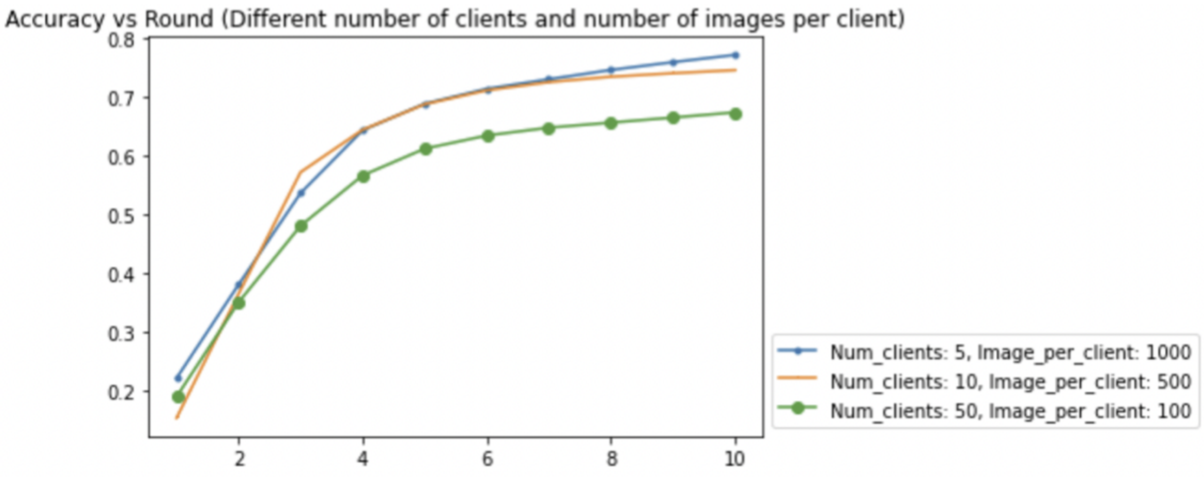


Figure 3.32: Validation accuracy of neural network trained with 5000 images in total

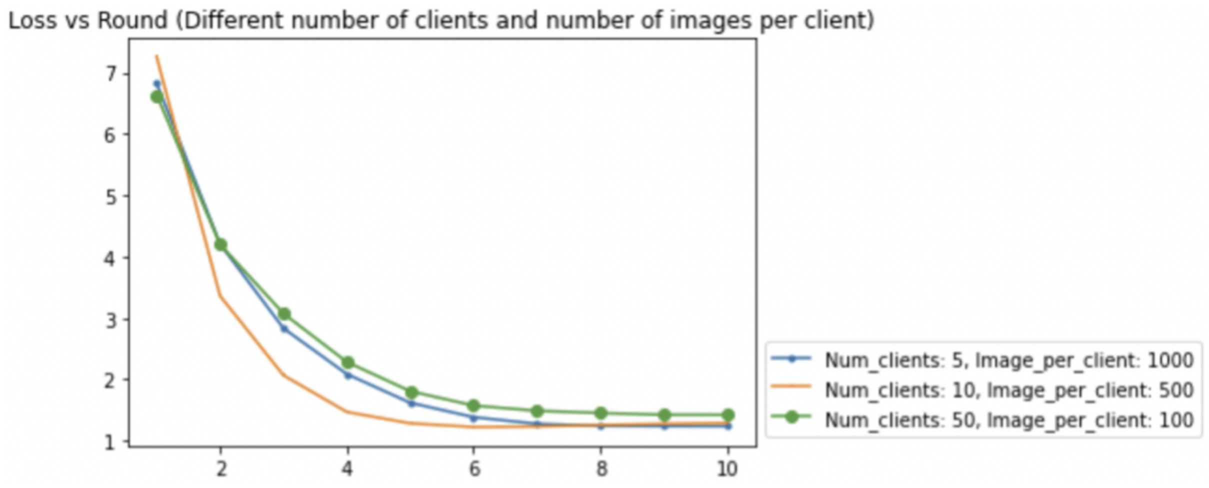


Figure 3.33: Validation loss of neural network trained with 5000 images in total

3.3 Sherpa AI

Sherpa.ai's Federated Learning and Differential Privacy Framework is a framework developed for research and experimentation in FL and Differential Privacy. It allows users to simulate the FL environment. Again, the effect of changing the number of clients and the number of data per clients are explored.

3.3.1 Fixed total number of data

MNIST is the dataset used in the experiments for this framework. In the first experiment, the total number of data is set to 3000, and the model is trained in 3 rounds.

Table 3.1: Fixed total number of data in Sherpa.ai

No. nodes	**No. data per client**	**Accuracy**	**Time (s)**
1	3000	0.8541	74.85
5	600	0.8371	206.66
10	300	0.7858	286.08
15	200	0.7495	461.30
20	150	0.5730	721.09
25	120	0.6433	903.46
30	100	0.5525	960.65

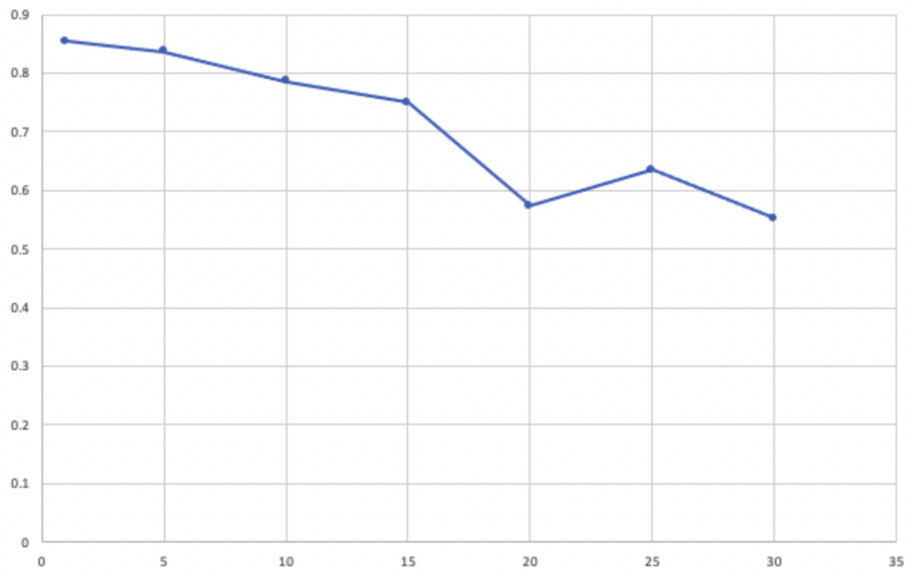


Figure 3.34: Accuracy of neural network trained with different numbers of nodes in Sherpa.ai

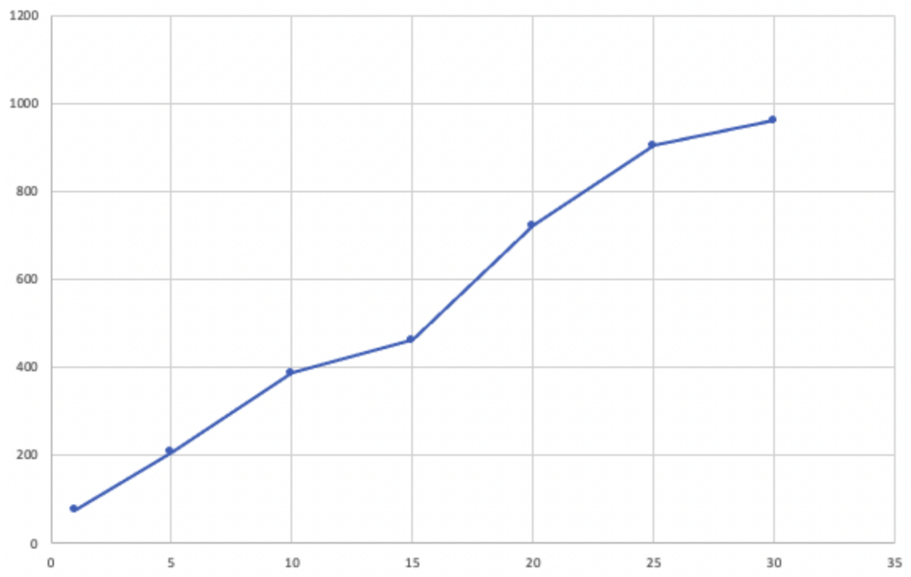


Figure 3.35: Time spent on training neural network with different numbers of nodes in Sherpa.ai

From Table 3.1 and Figure 3.34, it is clear that the accuracy of the neural network decreases when the number of clients increases, and when the number of data per client decreases. This is the same as the result we get from using Tensorflow Federated. Also, from Figure 3.35 it can be seen that the time spent on the training increases when there are more clients, because more communications and computations are required to train the network with data from multiple clients.

3.3.2 Fixed data per client

In this section, the number of data per client is fixed to be 300. Again the model is trained in 3 rounds.

Table 3.2: Fixed number of data per client in Sherpa.ai

No. nodes	**No. total data**	**Accuracy**	**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

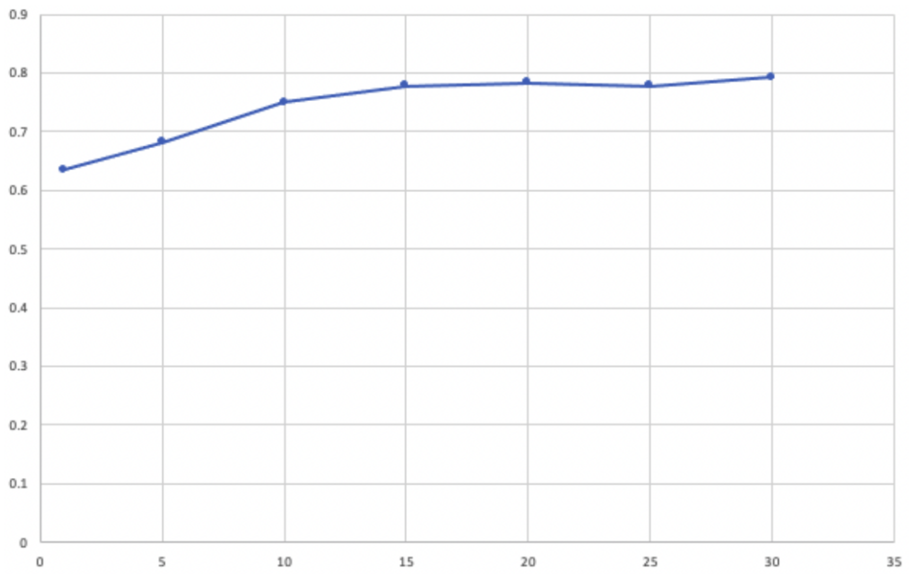


Figure 3.36: Accuracy of neural network trained with different numbers of data per client in Sherpa.ai

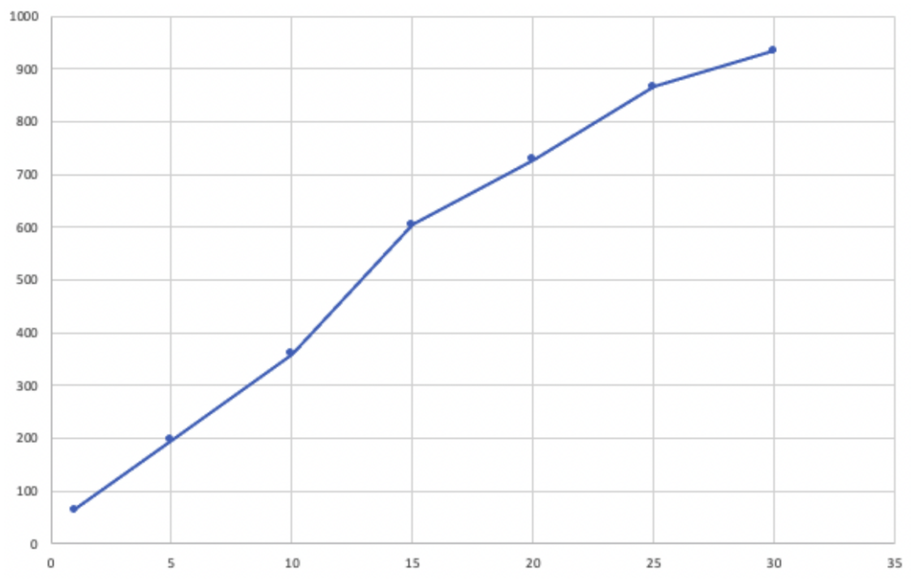


Figure 3.37: Time(s) spent on training neural network with different numbers of data per client in Sherpa.ai

From Table 3.2 and Figure 3.36, we can see that the accuracy of the neural network increases with the number of clients. However, this is because the total number of data also increases with the number of clients, so that the neural network is trained on more data when there are more clients. The time spent on training the neural network increases with the number of clients as expected.

3.4 Flower

Flower is another python framework for federated learning. We implemented a simple server-client platform using Flower. In each round, the server selects specific clients, and sned the updated model to the selected clients. The clients only need to connect to the server and wait for the server to select them. When the client receives the model, it trains the model locally with its local data, then the updated gradients are sent back to the server for aggregation.

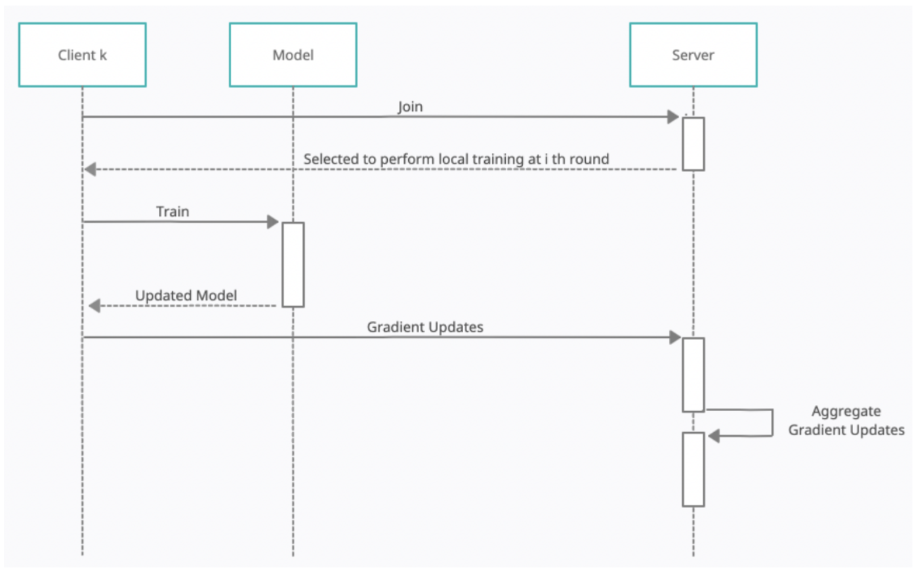

Client k
Model
Server
Join
Selected to perform local training at i th round
Train
Updated Model
Gradient Updates
Aggregate Gradient Updates

Figure 3.38: Sequence Diagram of one round in the FL CLI platform

3.4.1 Rationale for using Flower

The Flower framework has 4 major advantages, scalability, compatibility with edge devices, proven infrastructure, and usability [6].

For scalability, Flower is designed for handling real-world problems with a large number of clients. Previously, researchers have tried to run tests with more than 10000 clients using Flower, so it should be easy to scale the platform.

Flower supports all kinds of servers and devices including Android, iOS, Raspberry Pi, and Nvidia Jetson. This allows it to be used in a lot more situations in the real-world, rather than being another research or experimental framework.

Flower handles all the communications and computations of federated learning, so users only need to focus on the model and the data when we use this framework. Users only need to type a few lines of codes to train their models using federated learning.

Finally, the codes of Flower is very easy to understand, and it provides a lot of freedom for customizing the federated learning algorithm to fit users' specific requirements.

3.4.2 Evaluation

Two datasets, MNIST and CIFAR10 are chosen for the experiments. The server selects 10% of the total clients connected to the server every round, given that there are at least 3 clients per round and at least 5 clients are connected to the server at any moment. Every client has 500 training samples and 10 test samples, and the aggregation strategy in the server is federated averaging. The tests are conducted locally on a single computer using the command line interface (CLI).

Table 3.3: Training result with the CLI platform using Flower

Dataset	**Model**	**No. of rounds**	**Accuracy**	**Time (s)**
MNIST	Fully Connected Neural Network (FCNN)	10	85%	24
CIFAR10	Convolutional Neural Network (CNN)	10	29%	106

In Table 3.3, the result is very similar to the result obtained using Tensorflow Federated. The neural network training on MNIST has a much higher accuracy than the one training on CIFAR10. This once again shows that CIFAR10 is a more complex dataset, because it contains coloured images while MNIST only contains black and white images, and it is a much more difficult task for computers to classify the CIFAR10 images. Another possible reason is that FCNN is a simpler network, whereas CNN is relatively more complex. Therefore FCNN converges more quickly than CNN. The differences in dataset and neural network also lead to the huge difference in the training time.

More experiments are conducted below to investigate the relationship between the number of clients, number of rounds, and the accuracy. The dataset used is MNIST. The accuracy increases with the number of rounds, and it also increases with the number of clients, because it is trained on more data.

Table 3.4: Training results with different No. of rounds with the CLI platform using Flower

No. of clients	**No. of rounds**	**Accuracy**
5	5	82.11%
5	10	85.50%
5	15	86.04%
5	20	86.47%

Table 3.5: Training results with different No. of clients with the CLI platform using Flower

No. of clients	**No. of rounds**	**Accuracy**
2	10	81.67%
3	10	84.15%
4	10	84.94%
5	10	85.50%

3.5 PySyft, PyGrid, and Syft.js

PySyft is a library developed for secure and private machine learning. It uses federated learning, and security measures such as differential privacy and encrypted computations. In previous sections, all of the experiments are conducted in a single machine locally, multi-machine training is yet to be implemented in the frameworks. In the next subsection a tool in PySyft called Duet will be introduced. It allows users to do multi-machine trainings easily.

3.5.1 Duet

Duet is a peer-to-peer tool in the PySyft library that allows two users to train a machine learning model collaboratively. In Duet, the server side is called the Data Scientist, whereas the client side is called the Data Owner. The data owner is the one with the training data, which is kept locally on his side, and not exposed to the data scientist side. The data scientist sends requests to the data owner to access and compute the training data, while the data owner have to decide whether he allows a certain operations to be performed on their data by the data scientist.

In fact, Duet is a perfect platform for small scale federated learning projects, for example, the COVID-19 detection project. It is not easy for all hospitals and government departments to be online and active at the same moment. It would be easier for AI InnoBio to train the model with these parties one by one, so Duet is a possible solution for them.

Below shows the result of training a FCNN model on MNIST dataset. First we only create one client who has all 60000 images. We connect the data owner to the data scientist and let the data scientist sends requests to the data owner to train the model. Even though Duet is designed for one data owner and one data scientist for each connection, we simulated multiple clients and multiple connections by dividing the training set into equal portions and train the model with the portions of data one by one. The results are similar to results from other frameworks, the more clients, the less accurate the model, because the total number of data is fixed, but they are separated into different clients.

Table 3.6: Training results with different No. of clients with Duet

No. of clients	**No. of data per client**	**No. of epochs**	**Accuracy**	**Time (s)**
1	60000	5	97.86%	1077
5	12000	5	96.71%	1268
10	6000	5	95.29%	1564
1	60000	10	98.00%	2156
5	12000	10	96.46%	2588
10	6000	10	94.48%	3139

3.5.2 PyGrid

To scale PySyft, PyGrid is required for deploying PySyft training plans. It allows PySyft training plans to be accessed from websites and other devices such as mobile devices and edge devices.

3.5.3 Syft.js

Syft.js is a frontend Javascript library for interacting with PySyft plans hosted in PyGrid. It is built on top of Tensorflow.js and it will allow users to add security protocols such as secure aggregation, differential privacy, and multi-party computation in the future.

3.6 Summary

Four open-source frameworks for Federated Learning have been tried out. A lot of experiments are conducted by changing the parameters such as the number of clients and the number of data per client. From the results, it can be concluded that when compared to models trained with traditional centralizing machine learning algorithm, models trained with federated learning has a lower accuracy given that they are trained on the same amount of data, and the same number of epochs. Also, more time is required in the training process because of the additional communications and computations.

Most frameworks are still being actively developed, and most of them does not support multi-machine federated learning. This leaves us with a limited choice of frameworks if we would like to build a federated learning platform to tackle real-world problems. The platform needs to be updated frequently as well, because the source codes of the frameworks change rapidly.

4 Challenges

4.1 New Technology

Federated learning is a relatively new technique, it has not been applied to many real-world applications, and most previous works are only for research purposes. There are relatively few references that this project can refer to, so we can only try our best to solve the obstacles using the limited resources, and in many cases, it is beyond our group members' capabilities.

4.2 Implementing Federated Learning Platform from scratch

Federated Learning for multiple clients and multiple machines is the project's ultimate goal. However, to implement such a system from scratch is far beyond our capabilities. A lot of networking issues, security issues, and issues related to the algorithms need to be tackled. Therefore, instead of implementing the whole platform from scratch, we explored several open-source frameworks and try to build on top of them.

4.3 Open-source Federated Learning Frameworks

The open-source federated learning frameworks we explored are the most popular ones. However, they are still being developed, and they are mostly for research and experimental purposes for now. For example, all of the frameworks we tried except PySyft do not support multi-machine federated learning, which is essential for building a working product that can be applied to real-world situations. Moreover, all of these frameworks are under active development. There are rapid changes in the source codes and how to use their APIs. This leads to a lot of outdated sources and outdated documentation. The official tutorials and documentation from these frameworks might not work on the latest versions. During this project, a lot of time is spent on looking at and understanding the source codes, as well as finding the matching versions for the frameworks.

5 Conclusion

Federated learning or decentralized machine learning will definitely have a huge impact in the future. Many industries can take advantages of this technology in the future even though it is still in an early development stage, and a lot more researches and testings are needed. Being able to train a machine learning model without gathering data to a single server or a single machine, and without exposing sensitive data to others is a great advantage. Companies and governments can make use of this to create powerful machine learning products to benefit the society.

In this project, we have done researches on different types of federated learning, and the architecture of horizontal federated learning. We have also tried out four open-source federated learning frameworks and successfully trained several neural networks on different datasets. We found out that the accuracy of the model trained using federated learning will be lower than that trained with the traditional centralizing machine learning method. In the future, we hope that a more stable and scalable multi-machine federated learning platform can be developed, so that more people can use it and create products that can help our society.

References

[1] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, “Federated learning for mobile keyboard prediction,” 2019.

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications.” ACM Transactions on Intelligent Systems and Technology (TIST) Volume 10 Issue 2, Article No. 12, January 2019, 2019.

[3] J. Konečný, B. McMahan, and D. Ramage, “Federated optimization:distributed optimization beyond the datacenter,” 2015.

[4] K. Hao, “How apple personalizes siri without hoovering up your data,” 2019.

[5] R. Amichay and R. Harash, “Israeli hospital trials super-quick saliva test for covid-19,” 2020. [Online; accessed 23-October-2020].

[6] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, “Flower: A friendly federated learning research framework,” 2021.

[7] J. Konečný, B. McMahan, and D. Ramage, “Federated optimization:distributed optimization beyond the datacenter,” 2015.