National University of Computer and Emerging Sciences

COVID-19 Detection using Federated Learning

Syed Muhmmad Ali Mustafa……18L-0926
Zeerak Ahmad……..……18L-0949
Hizafa Nadeem……..…18L-1105
Kainat Asif……18L-2115

Supervisor: Dr. Zareen Alamgir

B.S. Computer Science
Final Year Project
April 2022

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:

Title: COVID-19 Detection using Federated Learning

is the sole contribution of the author(s) and no part hereof has been reproduced on **as it is** basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: 06/04/2022

Student 1

Name: Syed Muhammad Ali Mustafa

Signature: 

Student 2

Name: Zeerak Ahmad

Signature: 

Student 3

Name: Hizafa Nadeem

Signature: 

Student 4

Name: Kainat Asif

Signature: 

Table of Contents

List of Tables

List of Figures

[1] Abstract

Constant endeavours are being made throughout the world, using artificial intelligence, to detect COVID-19 after its spread. To ensure user's data privacy and consume less computational resources, COVID-19 can be detected with the help of Federated Learning instead of traditional Machine learning. The Federated Learning model will be trained on publicly available chest x-ray datasets. After training, the model will enable determination of positive or negative COVID-19 through the x-ray of the patient's chest. We will perform research and acquire knowledge about models and datasets available from sources such as published research papers, with the goal of combining the best of each to create an optimised model suitable for training and testing. The architectural strategy used is client-server architecture. The system will consist of a server and several clients that will be the hospitals. The server will send the global model to the clients. Clients will update their local models and clients will send gradiets. The gradients are then collected and aggregated by the server and the global model is updated. To check if a patient is COVID positive, the user of the system will upload a chest x-ray image into the system. The system will first preprocess the image before passing it to the model and the model's output is returned. The output of the system will specify whether the patient has Covid or not. In order to implement the system, we will be using Python notebooks. A notebook will be created for the server and each client will be represented by another notebook. The client's data will be stored in a file system having separate folders for train and test data. For each train and test data folder, there will be a "COVID" and "No Findings" folder to contain the x-ray images. For simplicity, we will begin with a synchronous system in which the server will wait to receive updates from each client before aggregating the gradients received. In the future, we will move to an asynchronous system. The privacy of the user will be ensured since no client-related data will be delivered to us, but the client will train our model on its data and only send the model gradients. Experiments were performed using two models, Darknet trained from scratch and covnext that is fine-tuned. The performance of both the models were similar but the performance of covnext is relatively high.

Chapter 1: Introduction

The current situation of Covid-19 has become a constant threat to the human life. AI and Deep Learning have been critical in identifying it based on recognized symptoms, including but not limited to cough, fever, and perhaps lung damage.

With that stated, machine learning models are being built, but they are encountering difficulties such as privacy concerns, since patient data is sensitive, and we cannot construct a good model without sufficient data [1]. That is when Federated Learning enters the picture. It is a machine learning approach introduced by Google in 2016 that enables the construction of models using remote datasets while maintaining data privacy [2].

Along with privacy, Federated Learning consumes less computational resources since it enables model training on client devices utilising locally stored models that are connected to a centralised global server that aggregates the dispersed model parameters of the client [3]. To maintain privacy and distribute computational power, every client prepares a local model independently using a sample and communicates its variables and weights with the global model.

1.1 Goals and Objectives

The project's main goal is to develop a system that will allow us to use federated learning to diagnose COVID-19 in patients. To accomplish that goal, we have outlined the following objectives.

- Collecting datasets (x-ray images, etc.) for training and testing purposes from publicly accessible sources
- Using the Python programming language and its tools to aid with data management and neural network construction
- Constructing a federated learning model using an environment that replicates a global model with many clients
- Throughout the process, ensuring user privacy
- Increasing accuracy through experimenting with various approaches and procedures

1.2 Scope of the Project

Federated learning, neural networks, and image classification are all included in the scope of this project. Although the project could be scaled up to diagnose a variety of other diseases using the same mechanism, we will be focusing on public datasets related to the COVID-19 epidemic.

Our datasets include chest x-rays from positive and negative COVID-19 patients.

Once the federated model is trained, we will be able to feed it chest x-rays and the model will determine whether or not the patient has COVID-19 disease.

COVID-19 is being detected via artificial intelligence. We will employ federated learning for its detection utilising images of x-ray of chest in order to protect user privacy and waste less computational power. We will implement the system using Python and CNNS. Once developed, the technology will be capable of taking x-ray images of chest and determining whether a patient is COVID-19 negative or positive. The report summarises a review of the

research on covid-19 identification via deep learning and federated learning. Prerequisites and The design of the project details the functional, non-functional, and hardware requirements, as well as the system architecture, use cases, design, and development phases. The implementation, in conjunction with the backend prototype, provides information on the model's implementation techniques and operation. At the conclusion of the report, the findings are summarised and a plan for future work is presented.

Chapter 2: Literature Survey / Related Work

The following sections describe the detailed literature survey of research done.

2.1 COVID-19 Detection

We examined many research articles grouped into two sections: one on using federated learning to covid 19 identification, and another on using deep learning for the identification of COVID 19.

COVID-19 is detected using deep learning techniques based on various of CNNs. To preserve privacy, Federated Learning models also are evaluated on the Chest X-rays dataset. The accuracy and performance of the FL model are evaluated to those of machine learning models.

1.1.2.1.1 COVID-19 Detection using Deep Learning

The identification of “COVID-19” is being investigated using various machine learning approaches/methods. “Deep Learning” is a highly effective classification and feature extraction. CNNs are used for COVID 19 infection detection in patients' chest x-rays. In order to detect the disease, a variety of CNN architectures are used. “S. Sanket, M. Vergin Raja Sarobin, and L. Jani Anbarasi et al.” [4] evaluated multiple variations folds of CNN models to detect “COVID 19” in x ray images of chest. [5] employed a “Dark Net model” with seventeen convolutional layers to filter and classify x-ray images. “R. Jain, M. Gupta, S. Taneja, and others” examined a variety of CNN designs. [6] employs “Inception V3”, “Xception”, and “ResNeXt” to detect “COVID-19” on the Chest X-rays dataset following preprocessing and data augmentation. “COVID” was detected in x-ray pictures using the approach in [7]. They conducted four stages of validation to determine the model's correctness. The model was tested and trained using a public dataset. Then they integrated the datasets from public and local sources for their training and testing. The third stage involved training and testing the model on a public dataset and a local dataset. Finally, the model was trained and evaluated on the local dataset using a combination of public and private datasets. “Antonios Makris, Ioannis Kontopoulos, and Konstantinos” [8] investigated “transfer learning” to mitigate the effect of limited datasets and big CNN architectures utilized to detect Pneumonia in COVID-19 patients. The authors of [9] created nCOVnet, a deep learning neural network approach for evaluating chest x-rays that were used as an extension of an existing study. “Bassi, P.R.A.S., and Attux, R.” combine “Twice Transfer Learning” with Output Neuron Keeping. [10] based on the National Institutes of Health's ChestX-ray14 dataset. Using criteria such as sensitivity, accuracy, kappa statistics, and specificity, “N. Narayan Das, N. Kumar, M. Kaur, V. Kumar, and D. Singh” [11] evaluated a CNN model based on the “Inception (Xception)” CNN. “A.K. Das, S. Ghosh, S. Thunder, and colleagues” [12] presented a Convolutional Neural Network by combining several states of CNN models - “DenseNet201”, “Resnet50V2”, and “Inceptionv3”. Rather than using a single perspective to access chest x-rays, the proposed approach in [13] processes x-rays from three separate angles and then integrates them. As a result, the model's precision is increased. [14] proposes the CovXNet deep neural convolutional network for extracting x-ray characteristics via depthwise convolution with different dilation rates. [15] employs CNNs to extract characteristics from x-rays, which are then put into a Support Vector Machine for image classification.

1.2.2.1.2 COVID-19 Detection using Federated Machine Learning

Through the use of "Federated Machine Learning", research has been conducted to detect COVID-19 via Chest X-Rays. "Federated Machine learning", as demonstrated by "Abdul Salam M, Taha S, and Ramadan M" [16], assures data privacy through centralised processing with high computation power. When the conventional machine learning model was compared to the "Federated Learning" model, it was established that the "Federated Learning" model had a low prediction loss and a high-performance time. "Trang-Thi Ho and Yennun-Huang" reported a privacy-preserving "Federated Learning" system for COVID-19 identification using chest X-ray images [17].

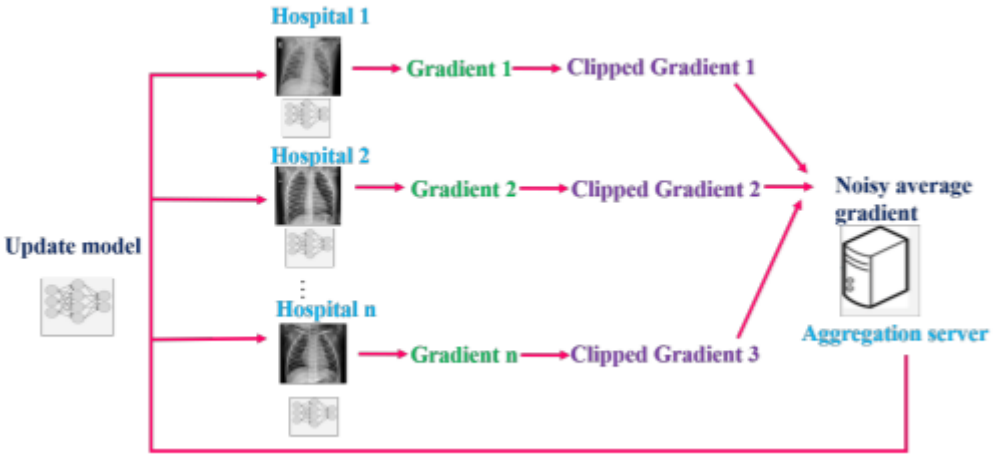


Figure 1: DPCOVID model [17]

Architecture for COVID-19 Detection

A decentralised model was introduced running across different hospitals without sharing patients' personal data. A strategy was also introduced for increasing accuracy for Non-IID COVID-19 data by increasing client fraction and computation per client. The "Federated Learning" model in the scope of decentralization was further explored by "Ines Feki, Sourour Ammar, Yousri Kessentini, Khan Muhammad" [18]. The model using "VGG 16" and "ResNet 50" CNN architecture was used to detect COVID-19 on the Chest Xray dataset. The specificity was identified to be trained on non-IID COVID-19 data. The model in [19] has combined a "Federated Learning" approach with "Transfer Learning" and compared the performance of pre-trained models "ResNet18", "ResNet50", "Deet121", and "MobileNetV2". The results show that "ResNet18" has the maximum accuracy. Then they compare "Federated Learning" with traditional machine learning. The performance of adaptive learning rate-based optimizers is also compared and "Momentum SGD" proves to have the best performance.

2.2 Literature Review Summary Table

Table 1: History of COVID-19 Detection Models

The summary of various COVID-19 Detection models created in the past from 2020-2021 is presented here.

No.	Name, reference	Inventor	Year	Input	Output	Description
1.	"Detection of novel coronavirus from chest X-rays using deep convolutional neural networks", [4]	"Sanket, S., Vergin Raja Sarobin, M.,Anbarasi Jani , L. et al"	2020	Chest Xrays images Accessed at: https://www.kaggle.com/tawsifurrahman/covid19-radiography-database	"X-ray image classified as COVID Positive, Normal"	"CovCNN" is a convolutional NN-based model for the analysis and detection of "COVID-19". "CovCNN" is composed of numerous CNN folds that are used to extract diverse features from "Chest X-ray images".
2.	"Automated detection of COVID-19 cases using deep neural networks with X-ray images", [5]	"Ozturk T, Talo M, Yildirim EA, Baloglu UB, Yildirim O, Acharya UR."	2020	Chest Xrays images Accessed at: https://github.com/muhammedtalo/COVID-19/tree/master/X-Ray%20Image%20DataSet	"X-ray image classified as COVID, No-Findings, or Pneumonia"	A "Dark Net" model was used that has 17 convolutional layers that filter x-ray images on each layer and classify the x-ray images in their respective class.
3.	"Deep learning based detection and	"Jain, R., Gupta, M., Taneja, S. et al."	2020	Chest Xrays images Accessed at:	"X-ray image classified as COVID, Normal or Pneumonia"	COVID-19 is detected using the Chest X-rays dataset utilising a variety of CNNs, including

	analysis of COVID-19 on chest X-ray images", [6]			thttps://www.kaggle.com/prashant268/chest-xray-covid19-pneumonia		"Xception", "Inception V3" and "ResNeXt". Following the application of preprocessing and data augmentation to the dataset. They also compared the accuracy of the CNN models.
4.	"Detection of COVID-19 from Chest X-ray Images Using Deep Convolutional Neural Networks", [7]	"Khasawneh N, Fraiwan M, Fraiwan L, Khassawneh B, Ibnian A."	2021	Chest Xrays images Accessed at: https://ieee-dataport.org/documents/covid-19-posteroanterior-chest-x-ray-fused-cpcxr-dataset	X-ray image classified as COVID and Normal	The model detected COVID in x-ray pictures using CNN. They assessed the model's accuracy in four steps. The model was initially trained and validated on a publicly available dataset. The dataset was then trained and tested using a combination of local and public resources. The third stage involved training the model on the dataset consisting and testing that on the localized dataset. Finally, the model has been trained and then tested on a combination of public and local datasets.
5.	"COVID-19 detection from chest X-Ray images using Deep Learning and	"Antonios Makris, Ioannis Kontopoulos, Konstantinos"	2020	Chest X-rays Images Accessed at: https://github.com/AntonisMakris/COVID1	"X-ray image classified as COVID, Pneumonia or Normal Predicted Class"	COVID-19 was detected using the CNN-based model on a 336-image X-ray dataset. "Transfer Learning" is being used to compensate for the dataset's short size. Pneumonia patients were

	Convolutional Neural Networks ”, [8]			9-XRay-Dataset		differentiated from COVID-19 patients using many large CNN architectures.
6.	“Application of deep learning for fast detection of COVID-19 in X-Rays using nCOVnet” , [9]	“Harsh Panwar, P.K. Gupta, Mohammad Khubeb Siddiqui, Ruben Morales-Menendez , Vaishnavi Singh”	2020	Chest X-rays Images Accessed at: https://arxiv.org/abs/2006.11988	“X-ray image classified as COVID-19 and Normal”	“nCOVnet”, a deep learning NN approach, is used as an extension of the existing study in the assessment of x-rays of chest.
7.	“A deep convolutional neural network for COVID-19 detection using chest X-rays”, [10]	“Bassi, P.R.A.S., Attux, R.”	2020	Chest Xrays Images NIH ChestX-ray14 database “COVID-19 image data collection” “(Cohen et al. (2020)), downloaded in October 2020”	“X-ray image classified COVID-19, pneumonia, and normal.”	Classification of the Chest Xray dataset is performed using a “Dense Convolutional Neural Network” with “Transfer Learning”. On “ImageNet”, the neural network is fine-tuned and pre-trained. The NIH Chest X-ray 14 dataset is used to use the “Twice Transfer Learning” approach. The retaining of output neurons is an enhancement to the twice transfer learning approach.
8.	“Automated Deep Transfer Learning-	“N. Narayan Das, N. Kumar, M. Kaur,	2020	Chest Xrays Images	“X-ray image classified COVID-19, pneumonia, or	CNN is being used to train the NN, similar to the “Inception (Xception)” model. Comparative Analysis is used to

	Based Approach for Detection of COVID-19 Infection in Chest X-rays", [11]	V. Kumar, D. Singh"			COVID negative"	evaluate the proposed model-based metrics' performance, including "accuracy", "f-measure", "sensitivity", "specificity", and "kappa statistics".
9.	"Automatic COVID-19 detection from X-ray images using ensemble learning with convolutional neural network", [12]	"Das, A.K., Ghosh, S., Thunder, S. et al."	2021	Chest Xrays Images from public datasets Accessed at: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia https://www.sirm.org/category/senza-categoria/COVID-19/	X-ray image classified COVID-19 or COVID negative	Multiple states of CNN models are acquired — "Resnet50V2", "DenseNet201", and "Inceptionv3". Combining CNN models using a weighted average ensemble technique is utilized to identify Chest Xray Images.
10.	"Triple-view Convolutional Neural Networks for	"Jianjia Zhang, Luping Zhou, Lei Wang"	2020	Chest Xrays Images from public datasets	X-ray image classified COVID-19, Normal or other	Rather than accessing chest x-rays from a single view, the proposed model processes x-rays from 3 different views, and then these 3 views are integrated. This

	COVID-19 Diagnosis with Chest X-ray", [13]			Accessed at: https://github.com/ieee8023/covid-chestxray-dataset		increases the accuracy of the model.
11.	"CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization", [14]	"Tanvir Mahmud, Md Awsafur Rahman, Shaikh Anowarul Fattah"	2020	Chest Xrays Images from public datasets Accessed at: https://data.mendeley.com/datasets/rscbjbr9sj/2	"X-ray image classified COVID-19, Normal, viral, or bacterial"	A deep neural convolutional network "CovXNet" that extracts x-ray features using depthwise convolution using varying dilation rates.
12.	"Detection of Coronavirus Disease (COVID-19) based	"P. K. Sethy, S. K. Behera, P. K. Ratha, and P. Biswas"	2020	Chest Xrays Images from public datasets	"X-ray image classified COVID-19, Normal or Pneumonia"	The model extracts features from x-rays using CNNs and feeds these features into a "Support Vector Machine" to categorize x-ray images.

	on Deep Features and Support Vector Machine", [15]					
13.	"COVID-19 detection using federated machine learning", [16]	"Abdul Salam M, Taha S, Ramadan M"	2021	Chest Xrays Images Accessed at: https://doi.org/10.1371/journal.pone.0252573.s001	X-ray image classified COVID-19 or COVID negative	"Federated Machine Learning" is a technique that is used to assure data privacy, centralised computation, and high computing capacity. On the basis of predictions loss and performance time, federated machine learning with traditional machine learning were compared.
14.	"DPCOVID: Privacy-Preserving Federated COVID-19 Detection", [17]	"Trang-Thi Ho, Yennun-Huang"	2021	Chest Xrays Images	"X-ray image classified COVID-19 or COVID negative"	"Federated Learning" is utilized to safeguard the hospital's data privacy by using a decentralized approach. On NonIID COVID-19 data, a technique was developed to increase the model's accuracy by increasing the number of clients, parallel (client fraction), and processing per client.
15.	"Federated learning for COVID-19 screening	"Ines Feki, Sourour Ammar, Yousri Kessentini, Khan	2021	Chest Xrays Images	"X-ray image classified COVID-19 or Normal Case"	FL framework for screening chest X-rays from multiple institutions while maintaining patient privacy. Determine the qualities and characteristics of data

	from Chest X-ray images", [18]	Muhammad"				distributions that are not identically distributed (non-IID) and are unbalanced.
16.	"Multi-diseases Classification from Chest-X-ray: A Federated Deep Learning Approach", [19]	"S. Banerjee, Misra, M. Prasad, E. Elmroth, and M. Bhuyan"	2021	Chest Xrays Images Accessed at: https://data.mendeley.com/datasets/rscbjbr9sj/2	X-ray image classified Viral Pneumonia, bacterial Pneumonia, or Normal class	The model has combined the "Federated learning" approach with "Transfer learning" and compared the performance of pre-trained models "ResNet18", "ResNet50", "Deet121", and "MobileNetV2". The results show that "ResNet18" has the maximum accuracy. Then they compare federated learning with traditional machine learning. Additionally, the efficiency of adaptable learning rate-based optimizers was compared, and "Momentum SGD" is found to perform the best.

1.3.

After the spread of COVID-19, a lot of research has been done to detect it using Artificial Intelligence. Federated Learning has its advantage over traditional Machine Learning in that it preserves users' data privacy and consumes less computational power. So, this study aims to develop a federated learning model that detects COVID-19 from chest x-ray images. The approach to be used is to do extensive research on the current work done on COVID-19 detection using Federated Learning and develop a hybrid model by combining the best-performing techniques.

Chapter 3: Requirements and Design

The sections below describe the system's requirements and design.

3.1 Functional Requirements

The following are the system's functional requirements:

1. The system shall allow the client to upload an image for diagnosis.
2. The system shall allow the client to train the model using his own local dataset.
3. The system shall allow the server to send its model to the client.
4. The system shall allow the client to receive an updated model from the server.
5. The system shall allow the server to receive updates (gradients, etc.) from the clients for aggregation.
6. The system shall display diagnosis results to the client.
7. The system shall allow the client to send updates (gradients, etc.) to the server.
8. The system shall allow the server to update the global model by aggregating the updates received from the clients.

3.2 Non-Functional Requirements

The following are the system's non-functional requirements:

1. The system (server) shall ensure reliability by having a 99 percent uptime.
2. The system shall ensure privacy by encrypting the model data before sending it.
3. The system shall provide an easy-to-use interface for the users to navigate through.
4. The system shall follow a consistent design theme.
5. The system shall handle scheduling through a queue system to avoid bottlenecks.
6. The system shall work with different image types.

3.3 Hardware and Software Requirements

The following are the system's hardware requirements:

- Cloud AWS - EC2 for server
- 8GB minimum RAM on client-side
- 1GBPS network port for the server
- Hardware firewall
- Consistent power supply

The following are the system's software requirements:

- Linux operating system
- Python (PyTorch) as the programming language
- Application programming interfaces (API) for client and server interaction
- Redis for queue management
- Filesystem interface for image storage on client-side
- SQL database on both client and server-side (used for version control as well)

3.4 System Architecture

The sections below describe the architecture of the system.

1.4.3.4.1 Internal Architecture

The internal architecture of the system consists of the following modules:

1. Client

   The client will retrieve the data, preprocess it, and then input it into the model. On the client's dataset, the client's local model will be trained. The server will then get the local model updates. The client will receive the server's global model updates and then update its local model. The client will make predictions based on the modified model. Clients will validate the model using labeled test data.

2. Server

   Global model weights will be initialized by the server. The server will receive local model changes and make global model updates. The server then notifies clients of global model updates. The server will monitor and tune the performance of the local clients.

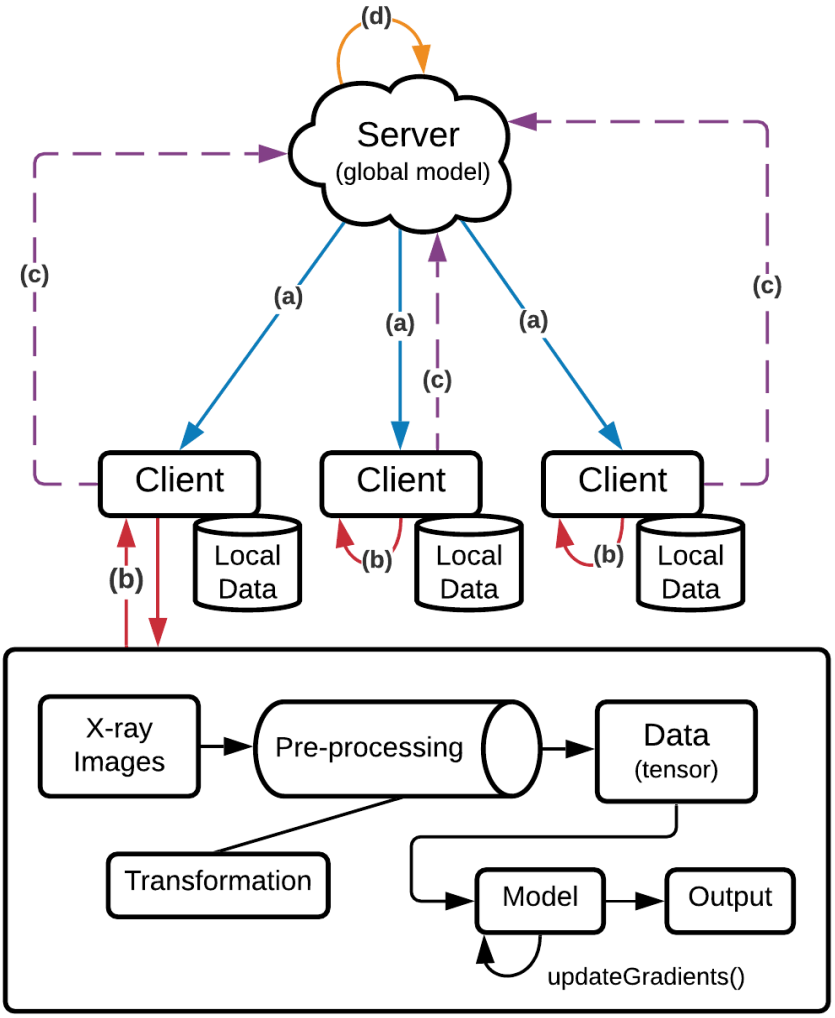


Figure 2: Internal Architecture

Client server internal architecture

a) Send updated global parameters

b) Local model updates

c) Send encrypted local gradients

d) Gradients aggregation and update global weights

1.5.3.4.2 External Architecture

The external architecture of the system is as follows:

Figure 3: External Architecture

Hospital and system external architecture

(a) Request client data

(b) Accept client data

(c) Send data to the system

(d) Receive output from the system

(e) Store results in database

The Chest X-ray Images are supplied into the system. The system pre-processes the images before giving it to the model. The model outputs whether the patient is impacted with COVID or not. This is the output of the system that is displayed to the user.

3.5 Architectural Strategies

Client-server architecture is the architectural strategy implemented. The client first obtains a copy of the global model from the server and trains it using local data. On the server-side, the server will receive global model gradients from clients, aggregate the received gradients, and update its global model.

A chest x-ray image is uploaded to the system to determine if a patient has COVID. The image is preprocessed before being fed into the model, and the model's output is returned to the user.

3.6 Use Cases

1.5.1.3.6.1 Upload an image

Name	Upload an image
Actors	Doctors, Nurses and lab assistants
Summary	Users will be able to upload an image.

Pre-Conditions	System should be working properly
Post Conditions	None
Special Requirements	None

Basic Flow			
Actor Action		**System Response**	
1	User will press the Upload Image button	2	System will ask to upload an image.
3	User will select an X-ray image and upload it.	4	System will classify the image into Covid or non-Covid.
Alternative Flow			
Actor Action		**System Response**	
3	User has uploaded an incorrect image.	4-A	System will generate a message: Incorrect image.

1.5.2.3.6.2 Train local model

Name	Train local model
Actors	Doctors, nurses and lab assistants
Summary	User will train model on it's local dataset
Pre-Conditions	Dataset must be uploaded
Post Conditions	None
Special Requirements	None

Basic Flow			
Actor Action		**System Response**	
1	User will press the training button	2	System will display the confirmation message
3	User will press on the confirmation button	4	System will train the model on the local dataset of the user.
Alternative Flow			
Actor Action		**System Response**	
3	User selected the cancel option	4-A	System will generate a message Action Cancelled .

1.5.3.3.6.3 Send updated model

Name	Send updated model
Actors	Doctors, nurses and lab assistants
Summary	Server will send the global model to the user
Pre-Conditions	System should be working properly
Post Conditions	None
Special Requirements	None

Basic Flow			
Actor Action		**System Response**	
1	User requests for the global model updates.	2	System will fetch the global model updates from the server.
Alternative Flow			
Actor Action		**System Response**	

1	User select the option do not update	2-A	System will generate a message: Not Updated.

1.5.4.3.6.4 Receive updated model

Name	Receive updated model
Actors	Doctors, nurses and lab assistants
Summary	User will receive the global model updates from server
Pre-Conditions	System should be working properly
Post Conditions	None
Special Requirements	Server must be available

Basic Flow

Actor Action		**System Response**	
1	User requests for the global model updates.	2	System will request the server to send global model updates.
3	User requests to display the updates	4	System will fetch the global model updates from the server and display the updates to the user.

Alternative Flow

Actor Action		**System Response**	
1	User selects the termination option	2-A	System will generate a message “Connection Terminated”.

1.5.5.3.6.5 Receive updates

Name	Receive updates
Actors	Doctors, nurses and lab assistants
Summary	Server will receive global model updates from the user

Pre-Conditions	System should be working properly
Post Conditions	None
Special Requirements	Server must be available

Basic Flow			
Actor Action		**System Response**	
1	User will click the send updates button.	2	System will make a connection with server
3	User will send updates (gradients).	4	System will send the updates to the server and the server will receive the updates for aggregation.
Alternative Flow			
Actor Action		**System Response**	
1	User clicks the cancel button.	4-A	System will generate a message “Action Terminated”.

1.5.6.3.6.6 Display results

Name	Display results
Actors	Doctors, nurses and lab assistants
Summary	System will display results of diagnosis
Pre-Conditions	System should be working properly
Post Conditions	None
Special Requirements	None
Basic Flow	

Actor Action		System Response	
1	User will click the Display button.	2	System will display the X-ray image and it's class label.
Alternative Flow			
Actor Action		**System Response**	
1	User clicks the cancel button.	2-A	System will generate a message "Action Terminated".

1.5.7.3.6.7 Update the global model

Name	Update the global model
Actors	Doctors, nurses and lab assistants
Summary	System will allow server to update the global model
Pre-Conditions	System should be working properly
Post Conditions	None
Special Requirements	Server must be available

Basic Flow			
Actor Action		**System Response**	
1	User will request for the global model updates	2	System will send request to the server to update the global model.
Alternative Flow			
Actor Action		**System Response**	
1	User clicks the termination option	2-A	System will generate a message "Connection Terminated".

1.5.8.3.6.8 Send gradients

Name	Send gradients

Actors	Doctors, nurses and lab assistants
Summary	User will send gradients on trained model to the server
Pre-Conditions	System should be working properly
Post Conditions	None
Special Requirements	Server must be available

Basic Flow			
Actor Action		**System Response**	
1	User will click compute gradients option	2	System will compute the gradients on the local trained model
3	User will click the send gradients option	4	System will send gradients to the server.

Alternative Flow			
Actor Action		**System Response**	
1	User clicks the termination option	2-A	System will generate a message “Connection Terminated”.

3.7 GUI

At the moment, we do not have a graphical user interface. Thus far, implementation has taken place via Google Colab on a Python notebook, with a particular emphasis on developing a model and structure to emulate how Federated Learning works.

3.8 Database Design

We will use the file system as our database to manage the datasets. Our model is distributed. Each client will be responsible for its own data usage. The only information required is chest x-ray images and a field indicating whether the chest x-ray image is positive or negative for COVID. To store such data, we will divide it into train and test folders. Each folder will contain two subfolders: "COVID" for chest x-ray images that are positive for COVID and "No Findings" for chest x-ray images that are negative for COVID.

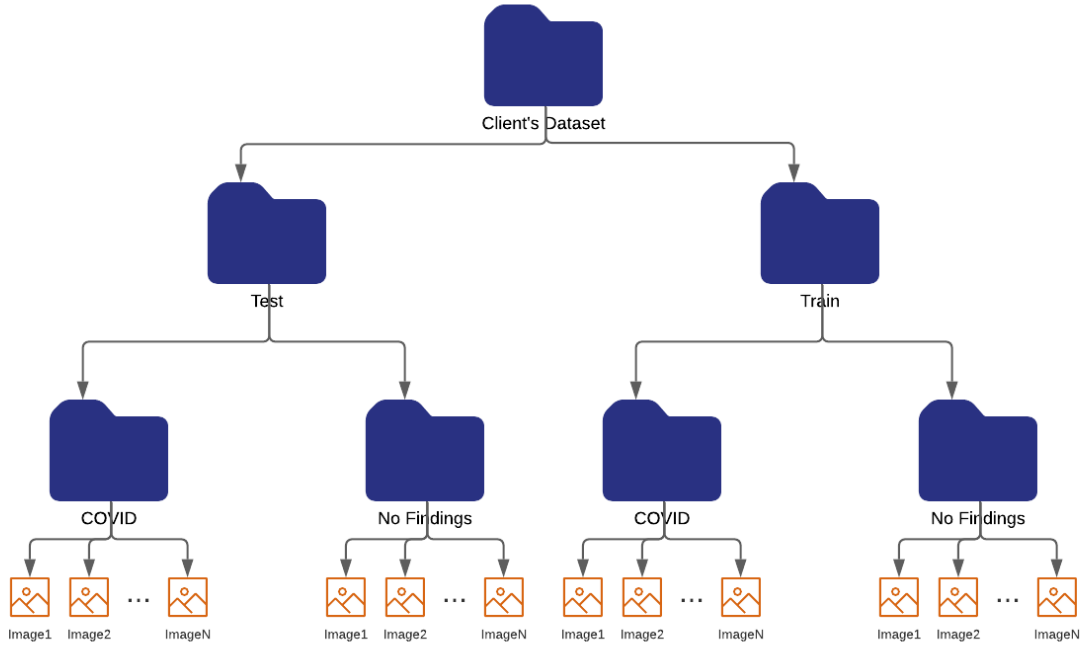


Figure 4: Database Design
Client's Database Design

3.9 System Requirements

Cloud AWS - EC2 hardware is required, as is a minimum of one gigabit per second network port for the server-side of the system. The server is in constant communication with the client, sending and receiving model updates. As a result, the server must be operational 24 hours a day. This requires a stable power supply. Additionally, we will install a firewall on the server-side to guard against any attacks on the global model. Due to the fact that clients must train their own local models, we require a minimum of 8GB RAM on the client-side.

In terms of software requirements, we will implement the system using Python on a Linux platform. The training and testing images will be stored on the client's file system, and we will provide an interface for client-server communication.

3.10 Design Considerations

To begin, we will synchronize the server and clients in order to train the model. This means that the server will wait to receive updated gradients from each client. Once the model is developed, we will upgrade our model to asynchronous. This means that rather than waiting for each client to update, the server aggregates the gradients after receiving updates from a specified number of clients. Additionally, instead of specifying a constant batch size for all FL iterations, the batch size may vary between iterations.

3.11 Development Methods

We will train the model using the distributed FL setting. Python notebooks will be used to develop both the client and server sides of the system. The server side will be implemented using a notebook, and the client-side will be implemented using a separate notebook. At the

moment, we're simulating a real-world scenario in which client notebooks serve as hospitals and server notebooks serve as servers.

We will distribute our dataset among clients, who will use it to train their own local models. Following model development in this environment, the system will be used in the real world, with clients being actual hospitals that train the models on their own data.

3.12 Class diagram

This section shows the class diagram of the system.

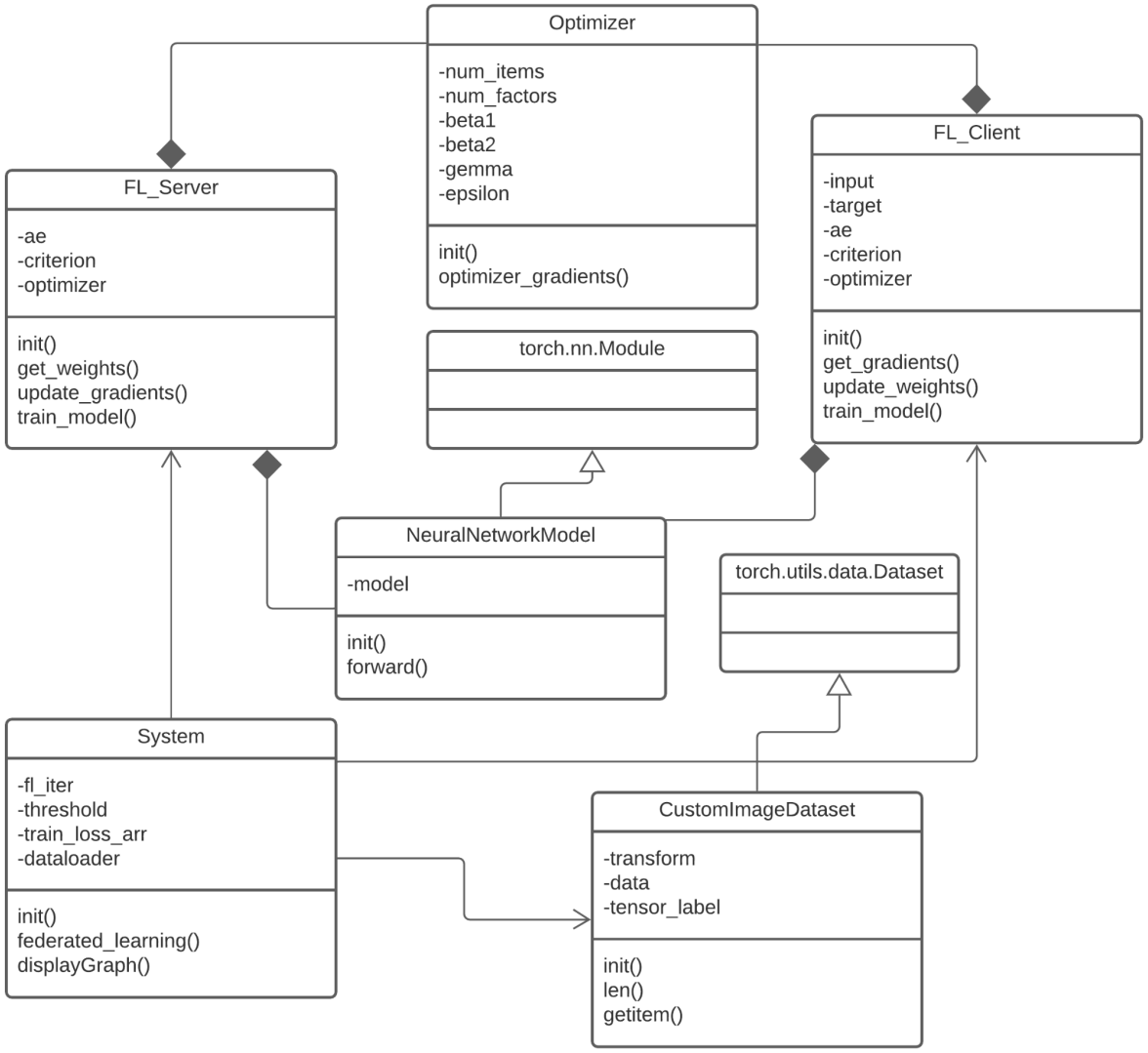


Figure 5: Class diagram
Class diagram of the system

3.13 Sequence diagram

The sub-sections below show the sequence diagram of the system.

1.6.3.13.1 Federated Learning

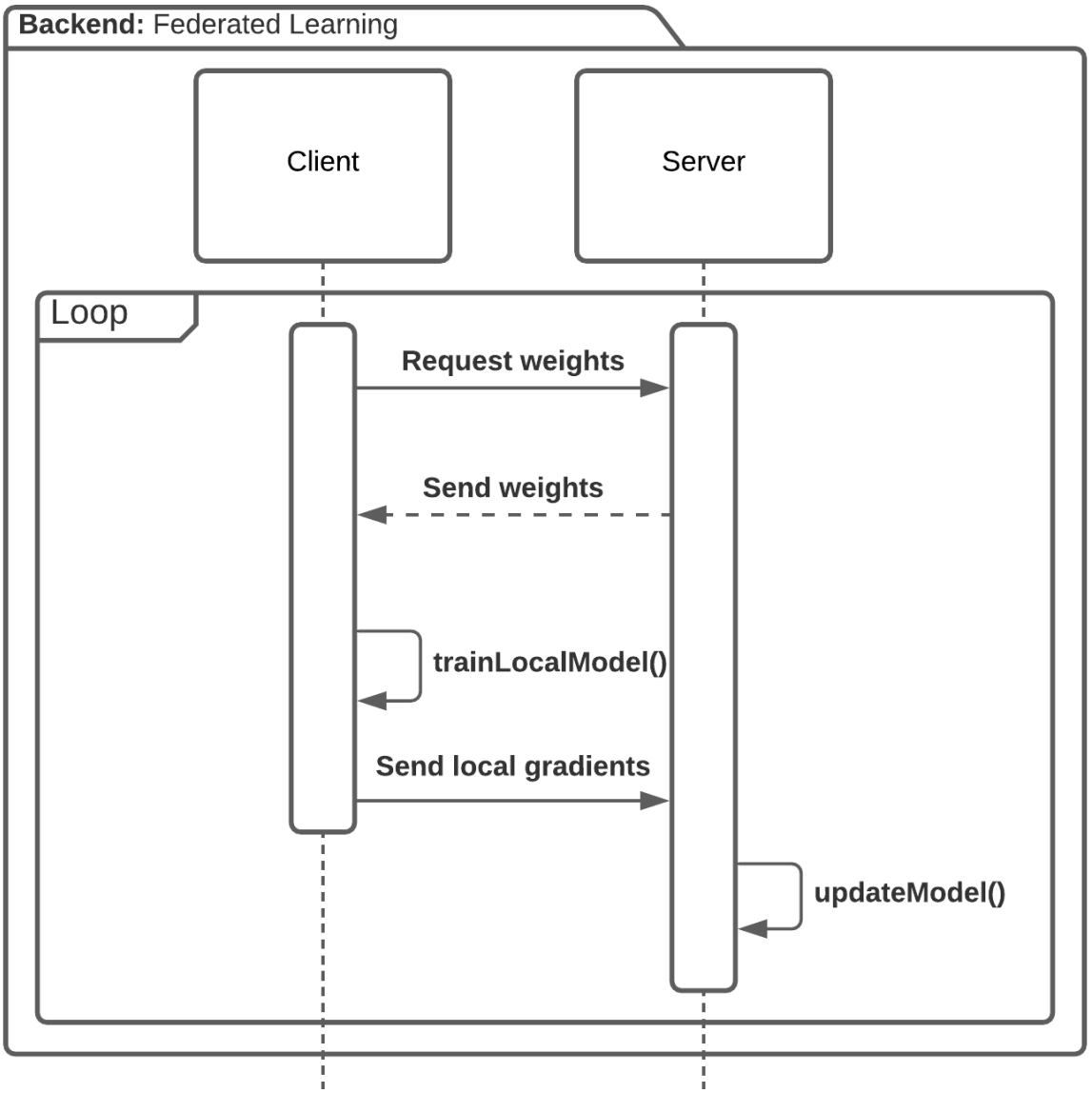


Figure 6: Sequence diagram 1
Sequence diagram on Federated Learning

1.7.3.13.2 COVID-19 Detection

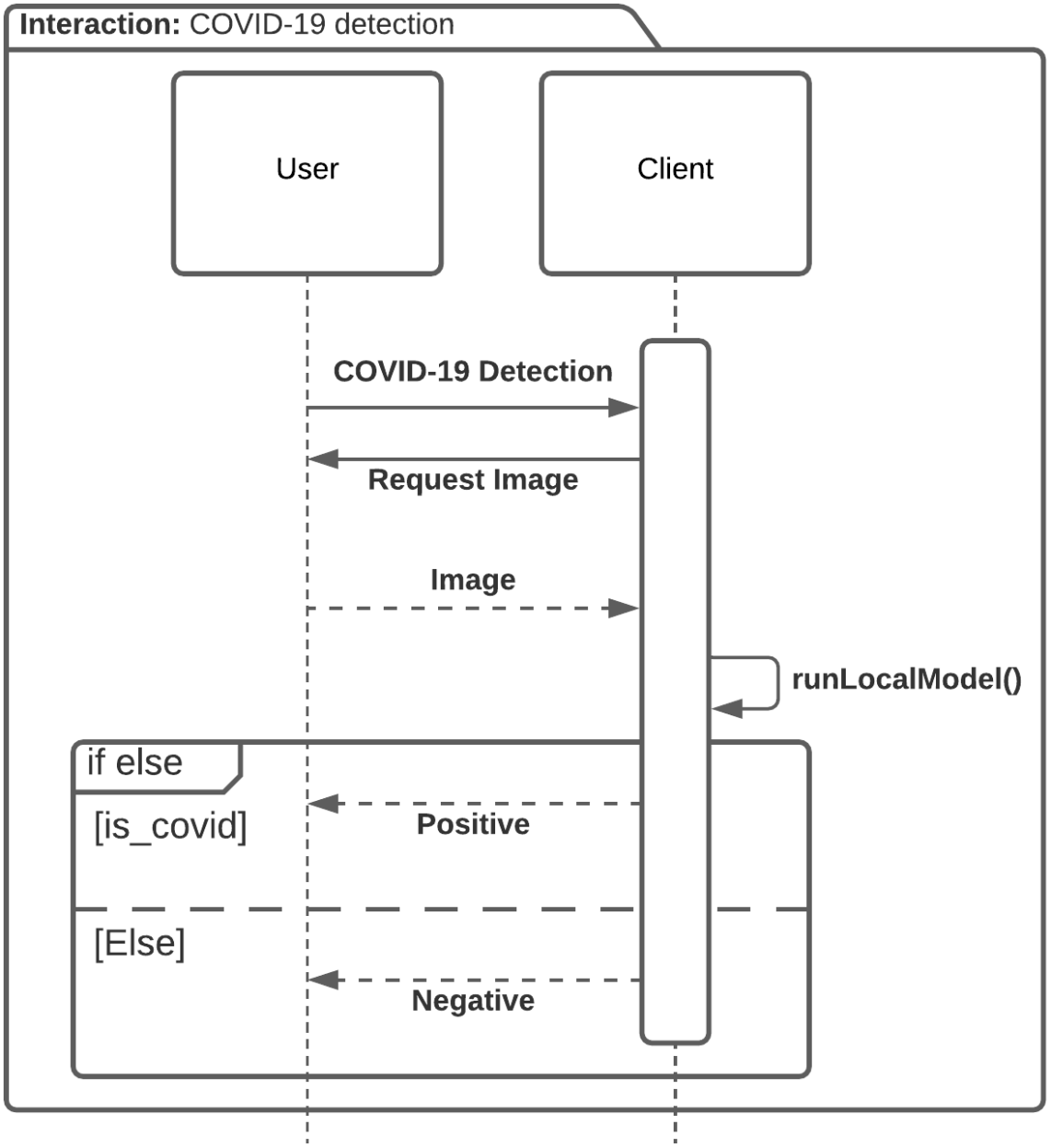


Figure 7: Sequence diagram 2
Sequence diagram on Covid-19 Detection

3.14 Policies and Tactics

For training the model with Federated Learning, our system will use a privacy-preserving policy. Our system will ensure client privacy by providing strong protection against information leakage by obscuring client data. Our system will not retrieve clients' private data; instead, updates to local models will be obtained, ensuring privacy and integrity. There is no data sharing permitted. Only authorized users will be able to upload and retrieve data from local devices. The mechanism of Federated Learning models ensures the confidentiality of the user's private data. The system will ensure transparency. The information presented to the user is concise and clear, with no disclosure of private or confidential information. Users will have access to the system, which will ensure consistency. Our system will be dependable and safe.

To conclude, the system must be capable of receiving images from the client and detecting whether the patient is positive for COVID. This is accomplished by training the model on the client's local data, aggregating and updating the global model. The system's non-functional requirements imply that it must be dependable, privacy-preserving, simple to use, consistent, and heterogeneous. The system architecture that will be employed is client server. A server and several clients will comprise the system. Clients will save their data locally in a file system. The system's server would be available 24 hours a day, and a firewall would be employed to secure the global model. Python will be used to implement the model. We will create an asynchronous system by utilizing Python notebooks for the server and each client. FL's method will safeguard the user's data's confidentiality.

Chapter 4: Implementation and Test Cases

We created a prototype backend implementation of Federated Learning using Google Colab. To simulate the process in that site, we use the Python programming language and the PyTorch Python library for creating and managing neural networks. Similarly, we imported additional libraries to perform additional functions such as image reading, graph display, and dataset management.

4.1 Implementation

This section highlights the implementation that has happened thus far.

4.1.1 Backend Prototype

The Python language was used in conjunction with the PyTorch Python libraries to create a simulation for Federated Learning in a single file by creating multiple classes such as Server and Client, as well as a custom Neural Network and Dataset class that would be used in a federated loop. We stored our training data in a Google Drive folder and mounted it on Google Colab for access.

When the file is first executed, it imports all libraries and mounts Google Drive storage to Google Colab. Following that, it imports a dataset from there and applies a series of transformations, as well as defining a label that will be used for loss calculation later. The dataset is then passed to a DataLoader class with a batch size of 32. Following that, the file defines classes such as Neural Network, Server, and Client. After that, our federated loop begins, but not before we create an object for our server and define some values for the loop, such as the threshold and federated loop iteration.

In each Federated Learning loop, the following steps repeat in a loop till a defined iteration limit.

1. Receive weights from Server and store them in a variable called global_gradients
2. Select random clients based on the threshold defined earlier
3. For each client, perform steps 4 to 10
4. Initialize local neural network
5. Set weights of local neural network to global_gradients
6. Pass train data and labels to the local neural network
7. Train the local neural network
8. Get gradients from local neural network and store them in a variable called local_gradients
9. Send local_gradients to the server
10. Go to step 4 for another client, if any.
11. Server passes all gradients received to an optimizer
12. Server then uses the gradients to calculate and update local weights

4.1.2 Client-Server Orchestration

As stated above, we created a simulation of Federated Learning by creating separate classes for client and server in the same file. Now, we have created a simulation of federated learning by implementing client-server architecture using python Django. The client is run through the python notebook and the server is implemented using Django and utilizes Redis for quick storage access. The environment is set up on the client-side. Endpoints are created to support communication between client and server. A Client uses one endpoint to send its local gradient to the server and fetch the global model from the server through another endpoint.

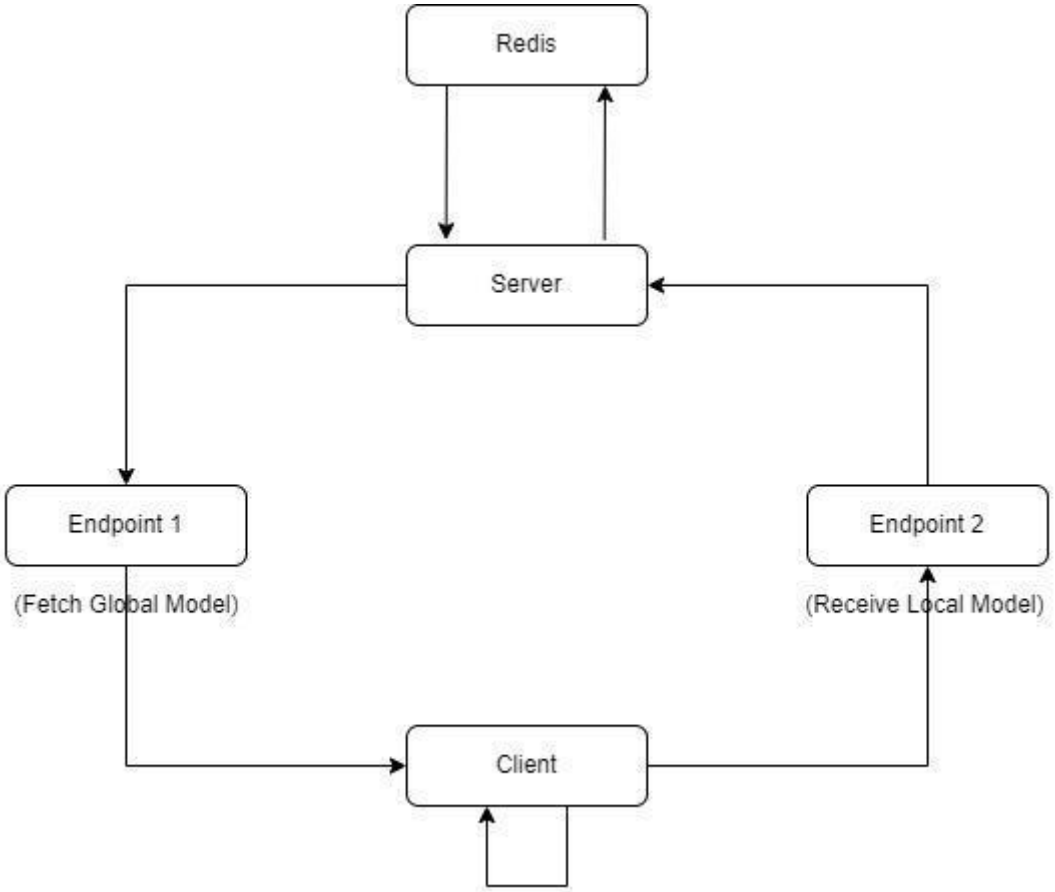


Figure 8: Client-Server Orchestration
Client-Server architecture diagram

The mechanism involved at both client and server-side for client-server orchestration are as follows:

Client-Side

Receives global model weights from the server. The client updates its local model by setting the weights of the local neural network to global_gradients. The client trains its local model and sends its gradient to the server in each epoch. When a gradient is sent it is converted into bytes then ASCII. Endpoints are created, one endpoint is used to receive global model weights and a second endpoint is used to send the local gradient of the client to the server.

Server Side

The server receives the local gradient from the client. The server stores the gradients on the Redis. When the server receives the local gradient from the client, it first checks if gradients are available in the Redis. The server fetches the stored gradients from Redis and aggregates them with local gradients. The aggregated gradient is saved on the Redis. When the client requests the global model updates, Server first fetches the computed gradients from the Redis and sends the gradients to the optimizer. Server updates and trains its global model. Once the global model is trained, it is saved and the server sends the global model to the client. The server serializes the global model before sending the model to the client.

4.2 Test Case Design and Description

4.2.1 Upload Image Test Case

Client Module			
Test Case ID:	*1*	**QA Test Engineer:**	*Ali Mustafa*
Test Case Version:	*1*	**Reviewed by:**	*Kainat Asif*
Test Date:	*07-04-2022*	**Use Case Reference(s):**	*Upload an image (1)*
Revision History:	*None*		
Objective:	*To check if user is able to successfully upload an X-ray image*		
Product/Ver/Module:	*Client module*		
Environment:	*The website is running and system is online*		
Assumptions:	*Upload button is visible to the user*		
Pre-Requisite:	*The user is already stored in the database and registered as a valid user*		
Step No.	**Execution description**	**Procedure result**	
1	*User clicks on the upload image button*	*System displays the file system.*	
2	*User selects the image from the file system and uploads it.*	*System displays the uploaded image.*	
Comments	The test case is passed. Our system is working according to our needs.		
☑ *Passed* ☐ *Failed* ☐ *Not Executed*			

4.2.2 Train Local Model Test Case

Client Module			
Test Case ID:	*2*	**QA Test Engineer:**	*Ali Mustafa*
Test Case Version:	*1*	**Reviewed by:**	*Kainat Asif*
Test Date:	*07-04-2022*	**Use Case Reference(s):**	*Train Local Model (2)*
Revision History:	*None*		
Objective:	*To check if local model is trained successfully on the user's machine*		
Product/Ver/Module:	*Client module*		
Environment:	*The website is running and system is online*		
Assumptions:	*Model training page is visible to the user*		
Pre-Requisite:	*The user is already stored in the database and registered as a valid user*		

Step No.	**Execution description**	**Procedure result**
1	*User clicks on the train model option*	*System redirects to the train model page*
2	*User clicks the train model button*	*System displays the training model symbol*
Comments	The test case is passed. Our system is working according to our need.	
■ *Passed* ☐ *Failed* ☐ *Not Executed*		

4.2.3 Send Updated Model Test Case

Server Module			
Test Case ID:	*3*	**QA Test Engineer:**	*Ali Mustafa*

Test Case Version:	*1*	**Reviewed by:**	*Hizafa Nadeem*
Test Date:	*08-04-2022*	**Use Case Reference(s):**	*Send Updated Model (3)*
Revision History:	*None*		
Objective:	*To check if the updated model is sent successfully to the user.*		
Product/Ver/Module	*Server module*		
Environment:	*The website is running and system is online*		
Assumptions:	*Server Dashboard is visible on the server side.*		
Pre-Requisite:	*The server is working and has a stable connection with the client.*		

Step No.	**Execution description**	**Procedure result**
1	*User clicks on the train model option*	*System redirects to the train model page*
2	*User clicks the train model button*	*System displays the training model symbol*
Comments	The test case is passed. Our system is working according to our needs.	
■ *Passed* ☐ *Failed* ☐ *Not Executed*		

4.2.4 Receive Updated Model Test Case

Server Module			
Test Case ID:	*4*	**QA Test Engineer:**	*Ali Mustafa*
Test Case Version:	*1*	**Reviewed by:**	*Kainat Asif*
Test Date:	*09-04-2022*	**Use Case Reference(s):**	*Receive Updated Model (4)*
Revision History:	*None*		

Objective:	*To check if the updated model is received successfully by the user.*	
Product/Ver/Module:	*Global Model module*	
Environment:	*The website is running and system is online*	
Assumptions:	*Model training page is visible to the user*	
Pre-Requisite:	*The server is working and has a stable connection with the client.*	
Step No.	**Execution description**	**Procedure result**
1	*User clicks on the train model option*	*System redirects to the train model page*
2	*User clicks the train model button*	*System trains the model and receive updated model.*
Comments	The test case is passed. Our system is working according to our need.	
☑ *Passed* ☐ *Failed* ☐ *Not Executed*		

4.2.5 Receive Gradients Test Case

Server Module			
Test Case ID:	*5*	**QA Test Engineer:**	*Ali Mustafa*
Test Case Version:	*1*	**Reviewed by:**	*Hizafa Nadeem*
Test Date:	*10-04-2022*	**Use Case Reference(s):**	*Receive Updates(5)*
Revision History:	*None*		
Objective:	*To check if the server has received gradients successfully from the user.*		
Product/Ver/Module:	*Server module*		
Environment:	*The website is running and system is online*		
Assumptions:	*Dashboard display is available on server side*		

Pre-Requisite:	*The server is working and has a stable connection with the client.*	
Step No.	**Execution description**	**Procedure result**
1	*User clicks send model updates*	*System redirects to the train model page*
2	*User clicks the send gradients*	*System displays the gradients received on the server dashboard.*
Comments	The test case is passed. Our system is working according to our needs.	
■ *Passed* □ *Failed* □ *Not Executed*		

4.2.6 Display Results Test Case

Client Module			
Test Case ID:	*6*	**QA Test Engineer:**	*Kainat Asif*
Test Case Version:	*1*	**Reviewed by:**	*Hizafa Nadeem*
Test Date:	*12-04-2022*	**Use Case Reference(s):**	*Display Results (6)*
Revision History:	*None*		
Objective:	*To check if results are displayed successfully on the system.*		
Product/Ver/Module:	*Client module*		
Environment:	*The website is running and system is online*		
Assumptions:	*Display Results page is visible to the user*		
Pre-Requisite:	*Server is working*		
Step No.	**Execution description**	**Procedure result**	
1	*User selected on the display results option.*	*System redirects to the results page.*	

2	*User clicks the display results button*	*System displays the results to the user.*
Comments	The test case is passed. Our system is working according to our need.	
☑ *Passed* ☐ *Failed* ☐ *Not Executed*		

4.2.7 Update Global Model Test Case

Server Module			
Test Case ID:	*7*	**QA Test Engineer:**	*Kainat Asif*
Test Case Version:	*1*	**Reviewed by:**	*Hizafa Nadeem*
Test Date:	*11-04-2022*	**Use Case Reference(s):**	*Update the global model (7)*
Revision History:	*None*		
Objective:	*To check if the global model is updated successfully on the server.*		
Product/Ver/Module:	*Global Model module*		
Environment:	*The website is running and system is online*		
Assumptions:	*Server Dashboard is available.*		
Pre-Requisite:	*The server is working and has a stable connection with the client.*		
Step No.	**Execution description**	**Procedure result**	
1	*User selects the train model option*	*System redirects to the train model page*	
2	*User sends the gradients to the server.*	*System updates the global model on the server.*	
Comments	The test case is passed. Our system is working according to our needs.		
☑ *Passed* ☐ *Failed* ☐ *Not Executed*			

4.2.8 Send Gradients Test Case

Client Module			
Test Case ID:	*8*	**QA Test Engineer:**	*Zeerak Ahmed*
Test Case Version:	*1*	**Reviewed by:**	*Kainat Asif*
Test Date:	*15-04-2022*	**Use Case Reference(s):**	*Send Gradients (8)*
Revision History:	*None*		
Objective:	*To check if the local model has successfully sent gradients to the server.*		
Product/Ver/Module:	*Client module*		
Environment:	*The website is running and system is online*		
Assumptions:	*Model training page is visible to the user*		
Pre-Requisite:	*The server is working and has a stable connection with the client.*		

Step No.	**Execution description**	**Procedure result**
1	*User clicks on the train model option*	*System redirects to the train model page*
2	*User clicks the train model button and prompts the server.*	*System sends the gradients to the server*

Comments	The test case is passed. Our system is working according to our need.
☑ *Passed* ☐ *Failed* ☐ *Not Executed*	

4.2.9 Performance Test Case

Performance			
Test Case ID:	*9*	**QA Test Engineer:**	*Zeerak Ahmed*

Test Case Version:	*1*	**Reviewed by:**	*Hizafa Nadeem*
Test Date:	*16-04-2022*	**Use Case Reference(s):**	*Performance*
Revision History:	*None*		
Objective:	*To check if the system is performing according to our expectations*		
Product/Ver/Module:	*Complete System*		
Environment:	*The system is running*		
Assumptions:	*The user knows about the functionalities of the system*		
Pre-Requisite:	*The user is logged in to the system*		

Step No.	**Execution description**	**Procedure result**
1	*The user performs an action*	*System responds quite fast*

Comments	The test case is passed. Our system is working according to our need.
☑ *Passed* ☐ *Failed* ☐ *Not Executed*	

4.2.10 Usability Test Case

Usability			
Test Case ID:	*10*	**QA Test Engineer:**	*Zeerak Ahmed*
Test Case Version:	*1*	**Reviewed by:**	*Kainat Asif*
Test Date:	*17-04-2022*	**Use Case Reference(s):**	*Usability*
Revision History:	*None*		
Objective:	*To check if the system is usable or not*		
Product/Ver/Module:	*Complete system*		
Environment:	*The system is running.*		

Assumptions:	*The user knows about the functionalities of the system*	
Pre-Requisite:	*The user is logged in to the system*	
Step No.	**Execution description**	**Procedure result**
1	*The user navigates through the system interface*	*The system responds accordingly*
Comments	The test case is passed. Our system is working according to our needs.	
☑ *Passed* ☐ *Failed* ☐ *Not Executed*		

4.2.11 Reliability Test Case

Usability			
Test Case ID:	*11*	**QA Test Engineer:**	*Ali Mustafa*
Test Case Version:	*1*	**Reviewed by:**	*Kainat Asif*
Test Date:	*17-04-2022*	**Use Case Reference(s):**	*Reliability*
Revision History:	*None*		
Objective:	*To check if the system is reliable or not*		
Product/Ver/Module:	*Complete System*		
Environment:	*The system is running*		
Assumptions:	*The user knows about the functionalities of the system*		
Pre-Requisite:	*The user is logged in to the system*		
Step No.	**Execution description**	**Procedure result**	
1	*The user performs an action*	*The system responds correctly with in the duration.*	
Comments	The test case is passed. Our system is working according to our needs.		
☑ *Passed* ☐ *Failed* ☐ *Not Executed*			

4.2.12 Privacy Test Case

Usability			
Test Case ID:	*12*	**QA Test Engineer:**	*Zeerak Ahmed*
Test Case Version:	*1*	**Reviewed by:**	*Hizafa Nadeem*
Test Date:	*18-04-2022*	**Use Case Reference(s):**	*Privacy*
Revision History:	*None*		
Objective:	*To check if the system provides user privacy or not*		
Product/Ver/Module:	*Whole system*		
Environment:	*System is in working state.*		
Assumptions:	*The user knows about the functionalities of the system*		
Pre-Requisite:	*The user is logged in to the system*		

Step No.	**Execution description**	**Procedure result**
1	*The user submits the data.*	*The system securely stores and transmits the user data using encryption.*
Comments	The test case is passed. Our system is working according to our needs.	
☑ *Passed* ☐ *Failed* ☐ *Not Executed*		

4.2.13 Consistency Test Case

Consistency			
Test Case ID:	*13*	**QA Test Engineer:**	*Zeerak Ahmed*
Test Case Version:	*1*	**Reviewed by:**	*Hizafa Nadeem*

Test Date:	*19-04-2022*	**Use Case Reference(s):**	*Consistency*
Revision History:	*None*		
Objective:	*To check if the system interface is consistent or not*		
Product/Ver/Module:	*Whole system*		
Environment:	*System is in working state.*		
Assumptions:	*The user knows about the functionalities of the system*		
Pre-Requisite:	*The user is logged in to the system*		
Step No.	**Execution description**	**Procedure result**	
1	*The user request a query*	*The system is available and responds to the user.*	
Comments	The test case is passed. Our system is working according to our needs.		
■ *Passed* ☐ *Failed* ☐ *Not Executed*			

4.3 Test Metrics

The test metrics are discussed below:

Metric	**Value**
Number of Test Cases:	13
Number of Test Cases Passed:	13
Number of Test Cases Failed:	0
Test Case Defect Density:	0/10
Test Case Effectiveness:	10/10

Traceability Matrix:	Included in a separate file

To summarize, we started with a simple backend “Federated Learning” model based on a “Machine Learning” model and then extended it to a model that served as a prototype for COVID-19 identification. Federated Learning was shown to be slower than Machine Learning, but it provides for greater privacy and the whole compute task can be distributed.

Chapter 5: Experimental Results and Analysis

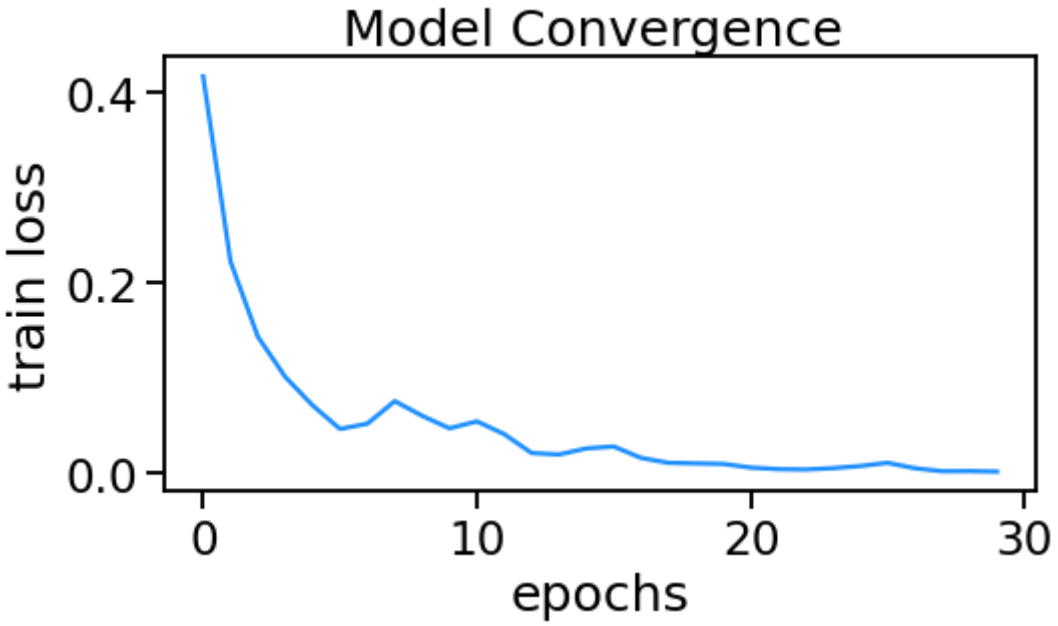
We have conducted experiments in different settings and environments to test the performances of our models. We have used 2 models DarkNet and Covnext. The first model Darknext was trained from scratch and the second model was Covnext pre trained on image-net dataset.

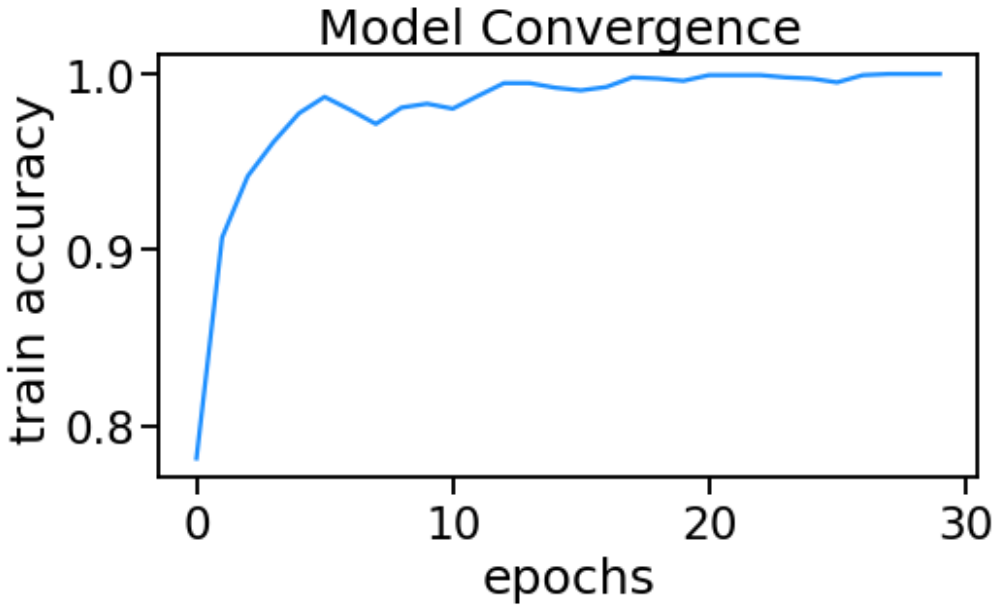
The models were trained using machine learning technique. Then federated learning technique was applied on both DarkNet and Covnext. We applied train test splits on the data set. The results were computed on both training and testing data. The evaluation metrics we used are “loss”, “accuracy”, “precision”, “recall” and “F1-score”.

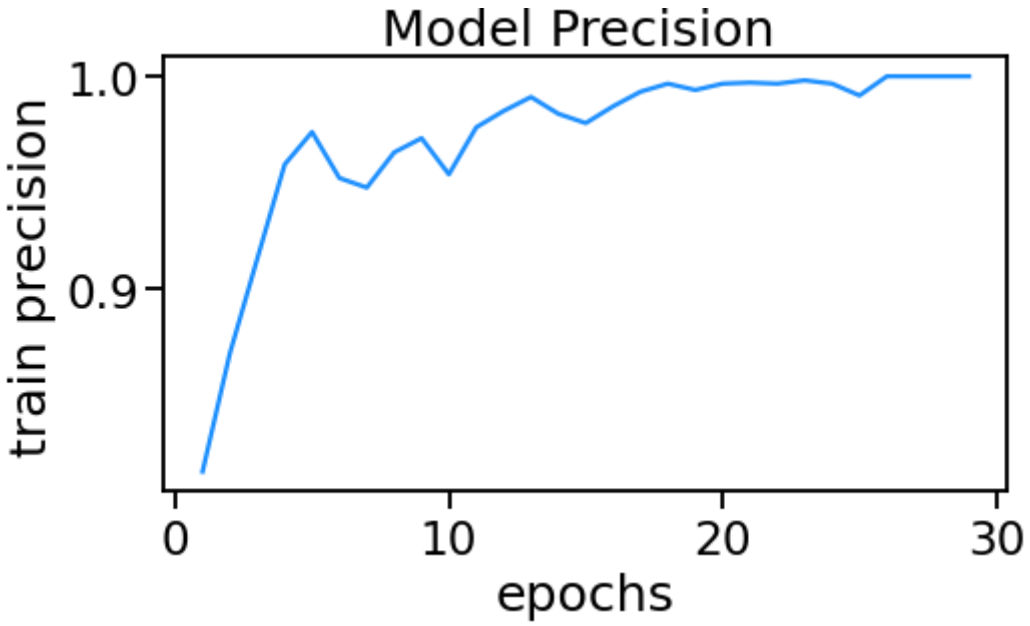
[2] 5.1 DarkNet

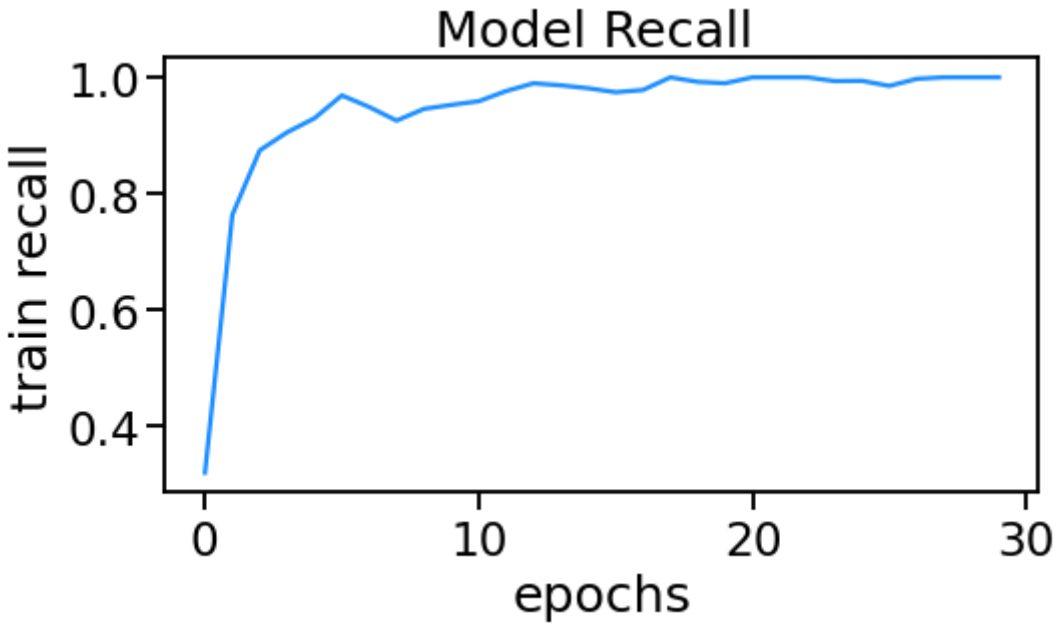
A model similar to DarkNet model was implemented for both machine learning and federated learning. The model was trained from scratch. The results of the DarkNet expreminent are shown below:

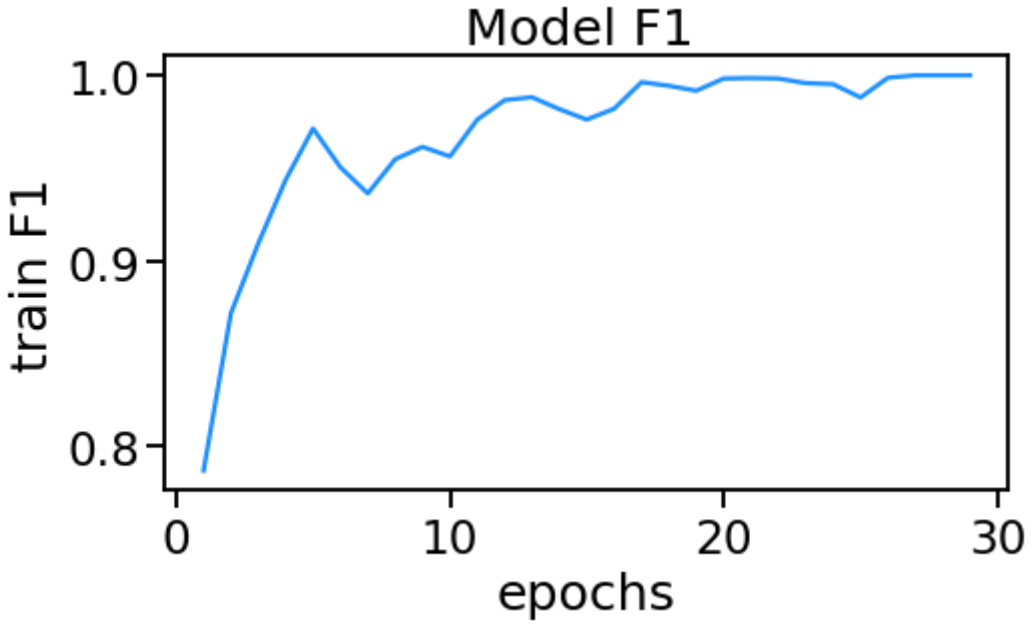
5.1.1 Machine learning




Model Convergence
train accuracy
1.0
0.9
0.8
0
10
20
30
epochs


Model Precision
train precision
1.0
0.9
0
10
20
30
epochs

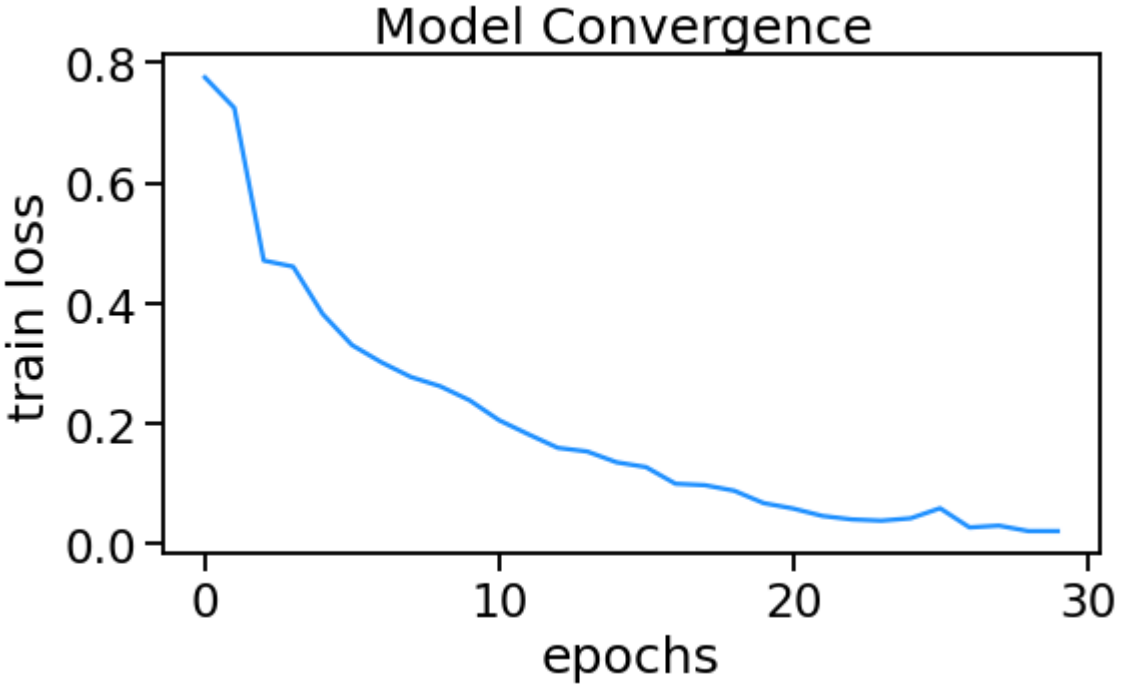


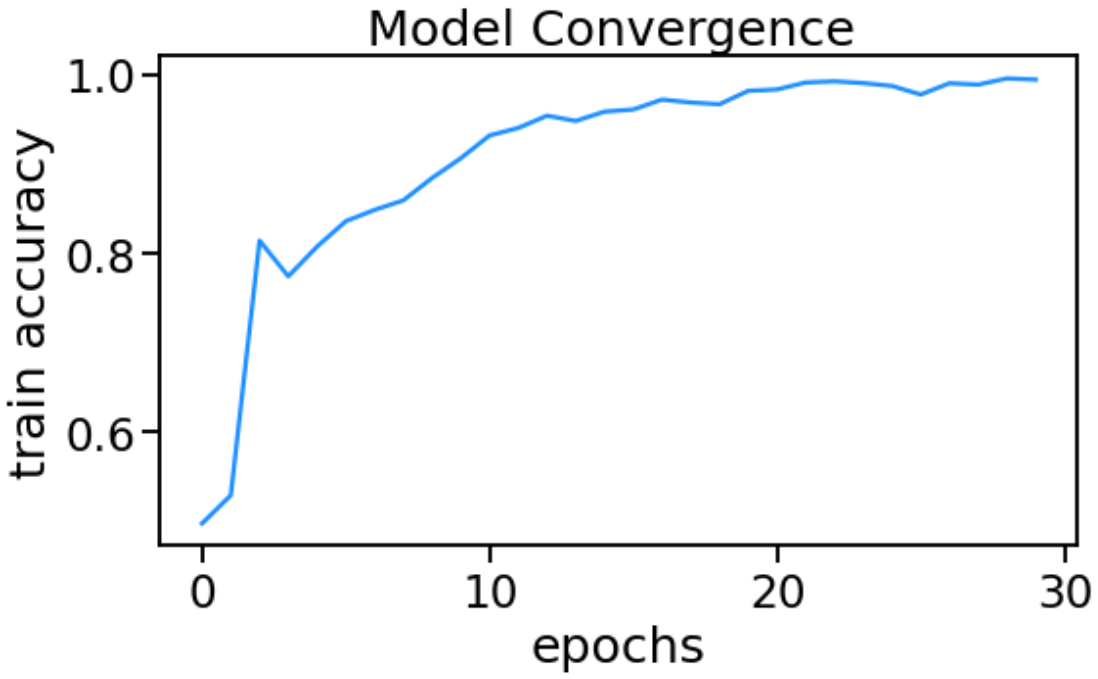


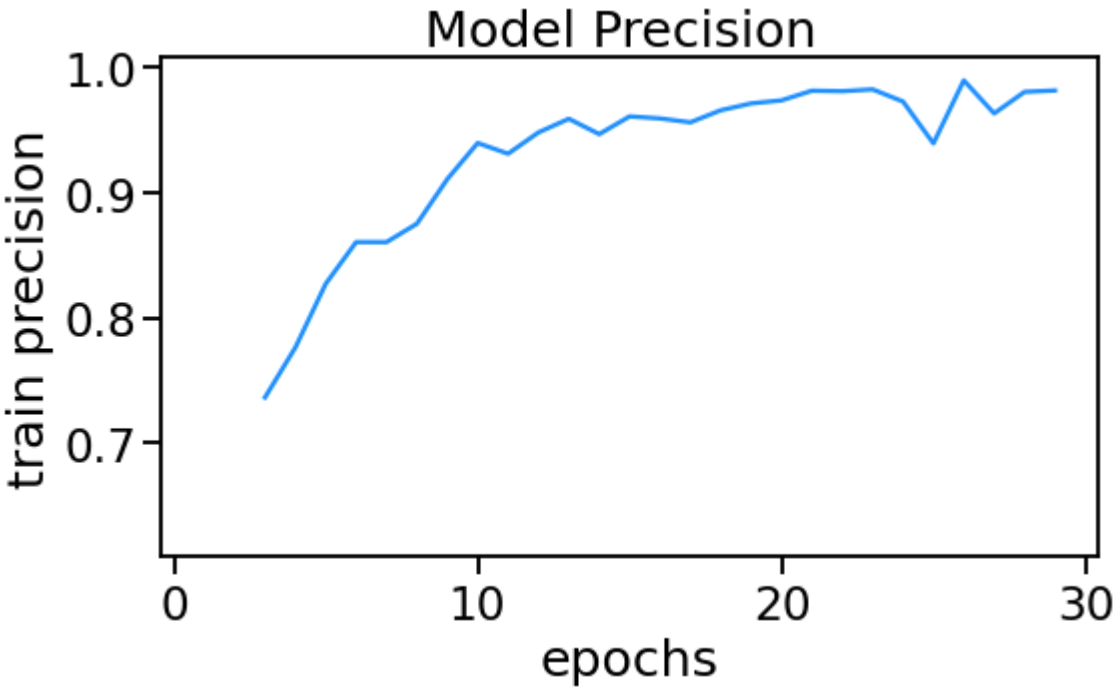
Test Results

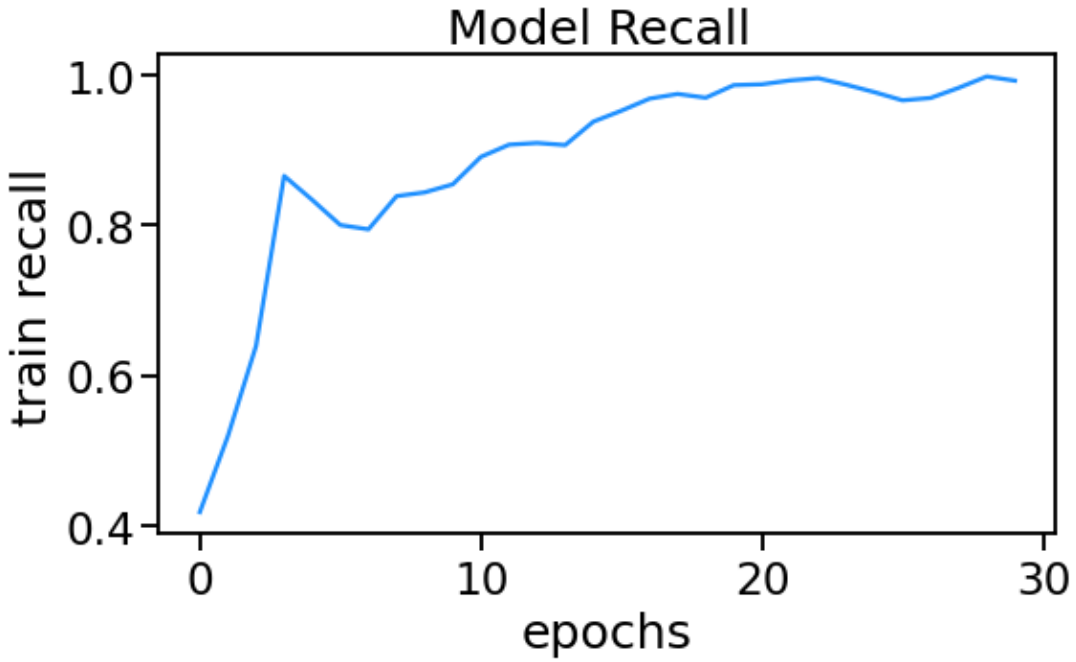
- Loss: 0.0013046801645677075
- Accuracy: 0.984375
- Precision: 0.9694444444444444
- Recall: 0.9436507936507935
- F1-Score: 0.9562850729517396

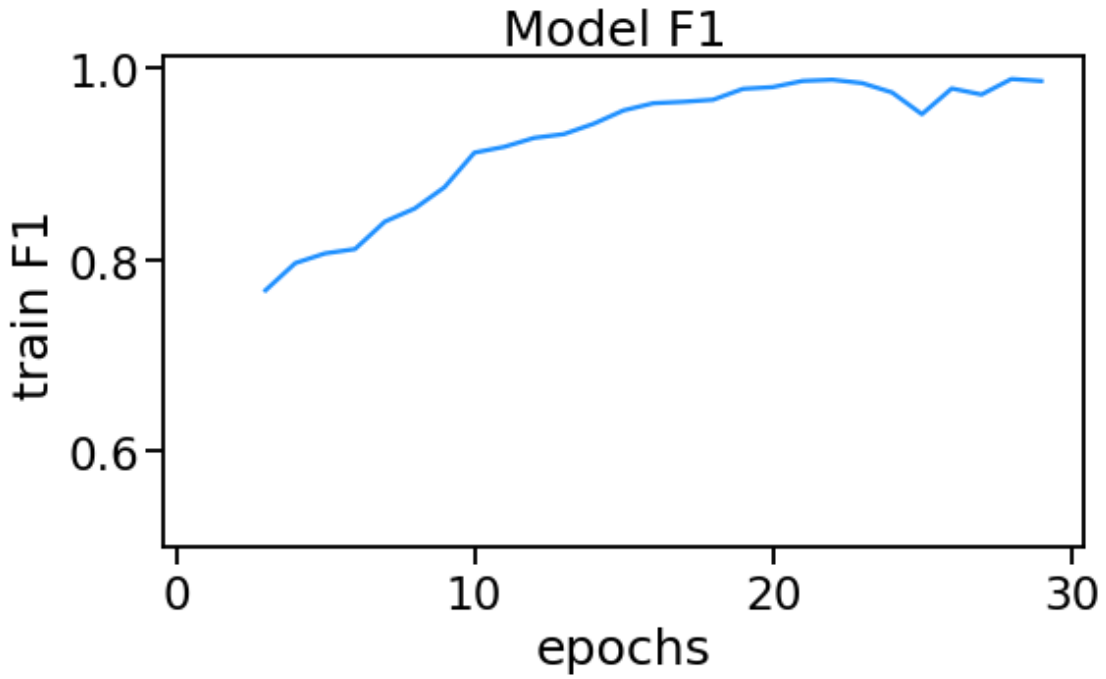
5.1.2 Federated learning






Model Precision
train precision
1.0
0.9
0.8
0.7
0
10
20
30
epochs


Model Recall
train recall
1.0
0.8
0.6
0.4
0
10
20
30
epochs



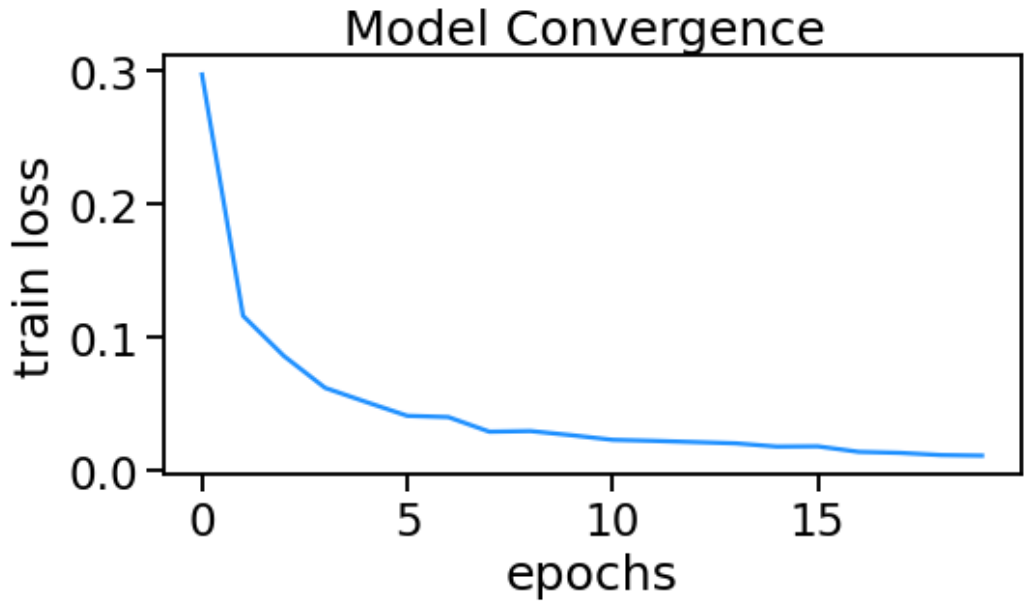
Test Results

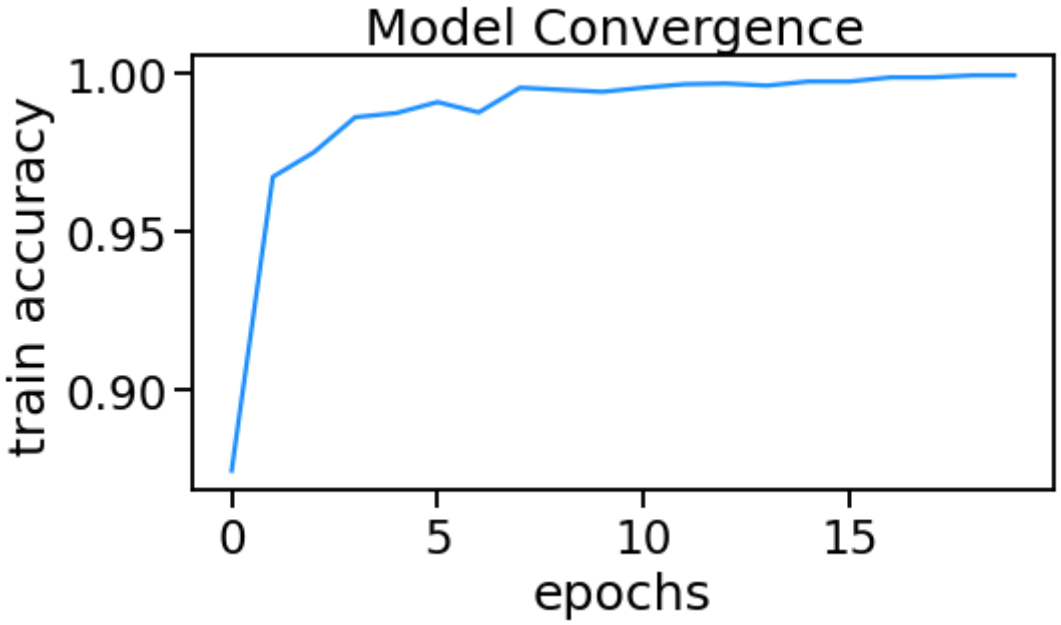
- Loss: 0.101012052192042 28
- Accuracy: 0.9605902777777778
- Precision: 0.8986202686202686
- Recall: 0.9442526455026455
- F1-Score: 0.9138210131545551

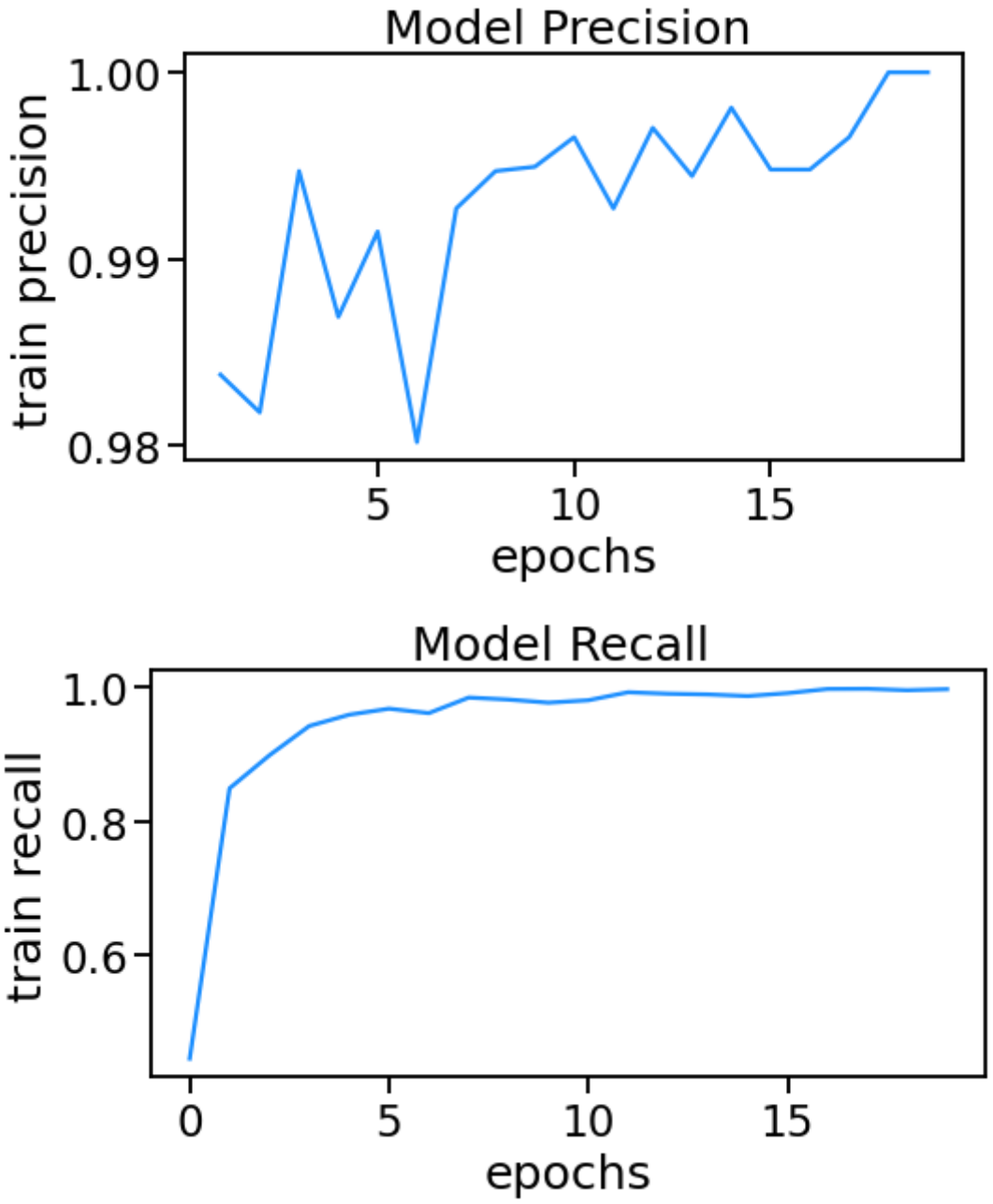
[3] 5.2 Covnext

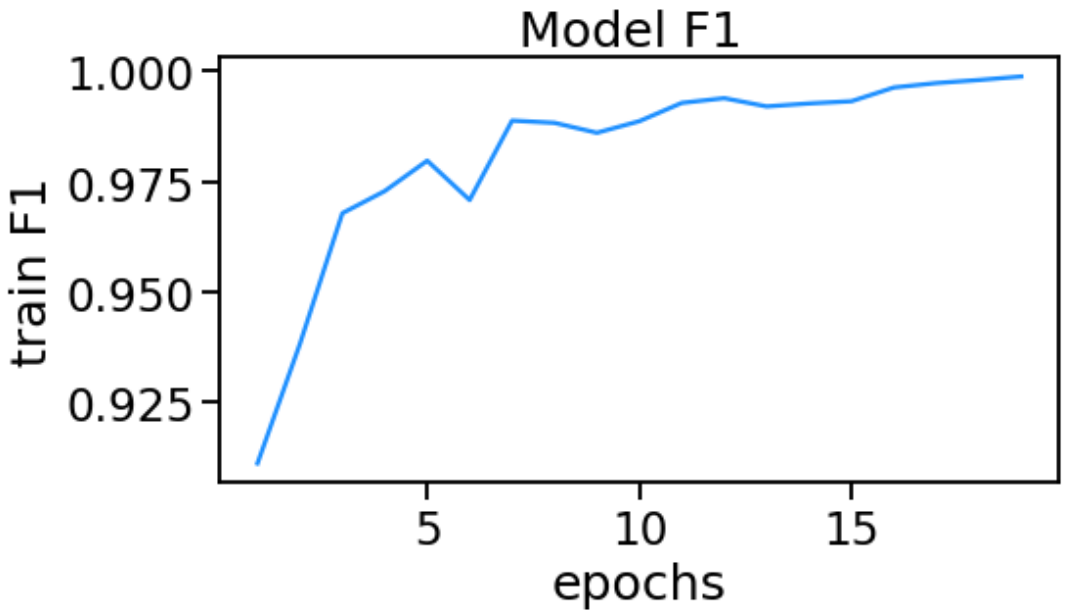
The Covnext model that is pretrained on Imagenet was used as a pretrained model. The model was fine-tuned to perform binary classification of COVID-19. The results of the Covnext expreminent are shown below:

5.2.1 Machine learning


Model Convergence
train loss
0.3
0.2
0.1
0.0
0
5
10
15
epochs


Model Convergence
train accuracy
1.00
0.95
0.90
0
5
10
15
epochs

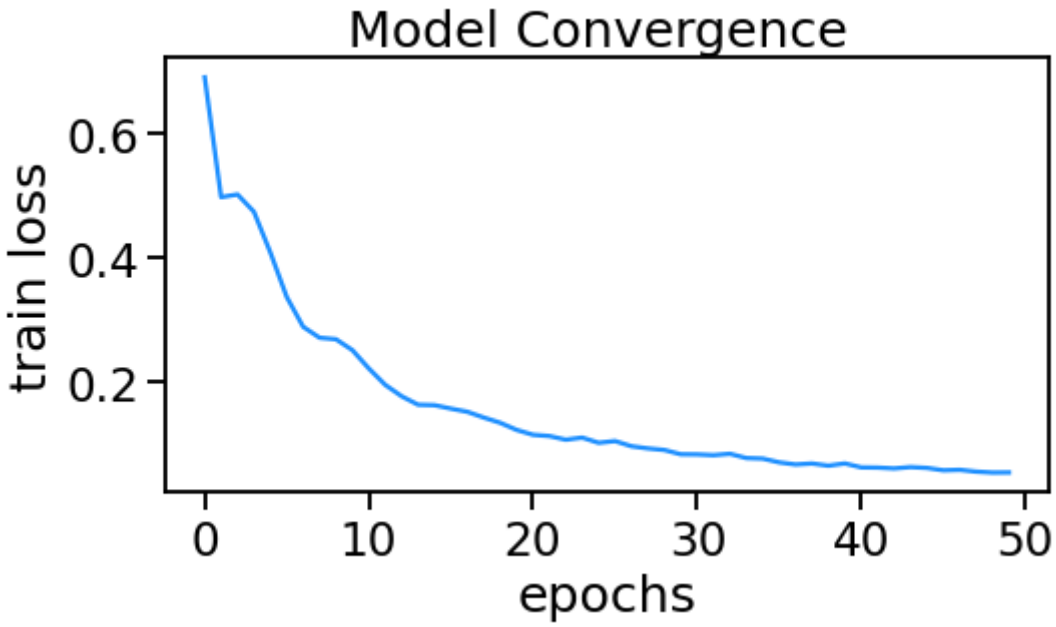

Model Precision
train precision
1.00
0.99
0.98
5
10
15
epochs
Model Recall
train recall
1.0
0.8
0.6
0
5
10
15
epochs

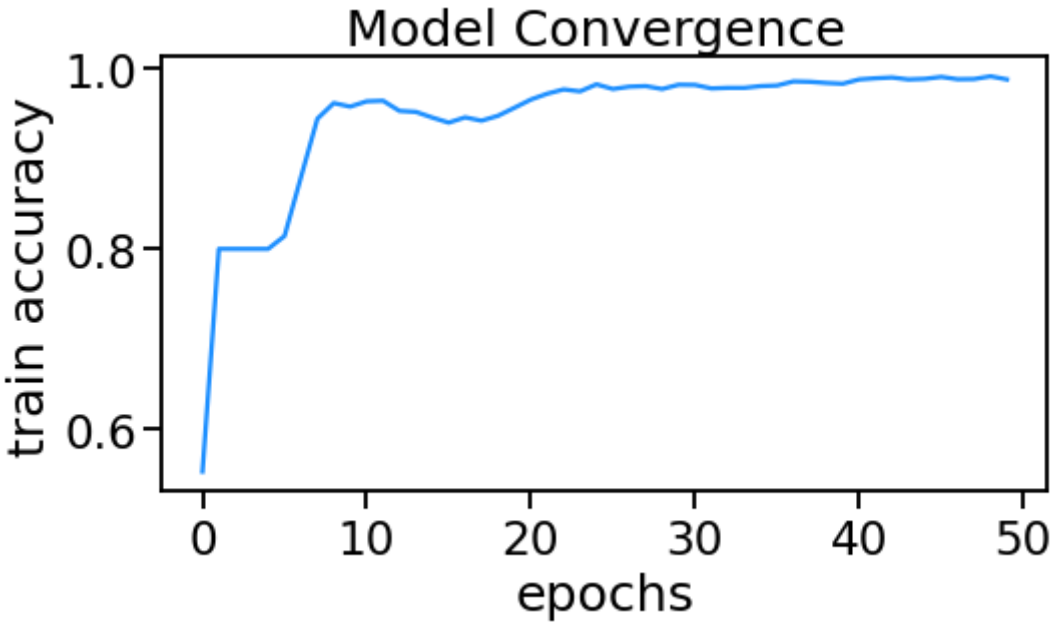


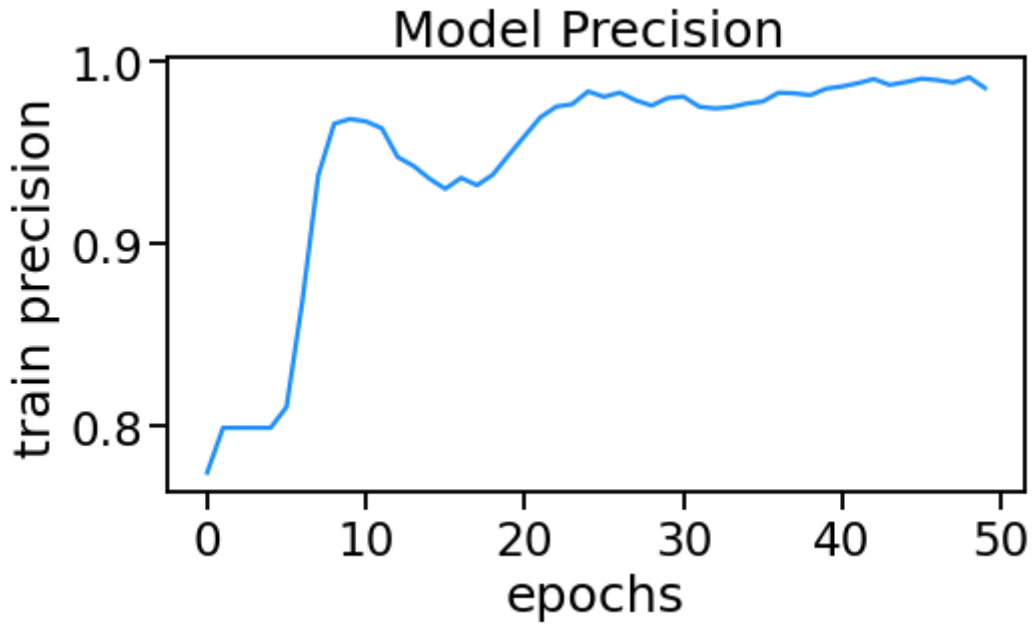
Test Results

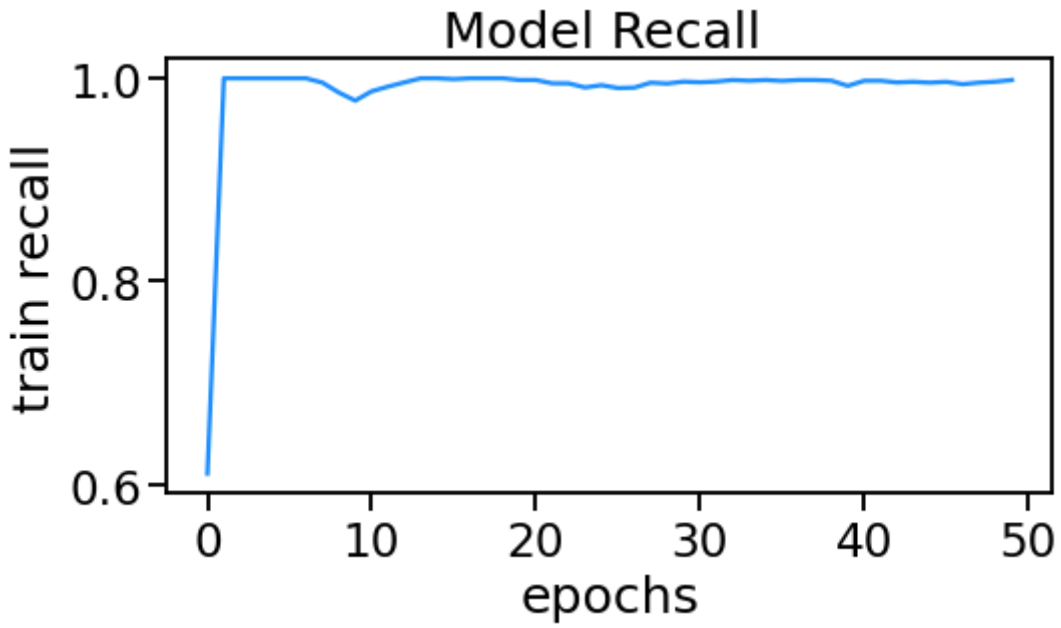
- Loss: 0.0013672651553023902
- Accuracy: 0.984375
- Precision: 0.9742063492063492
- Recall: 0.9317460317460317
- F1-Score: 0.9505450926139657

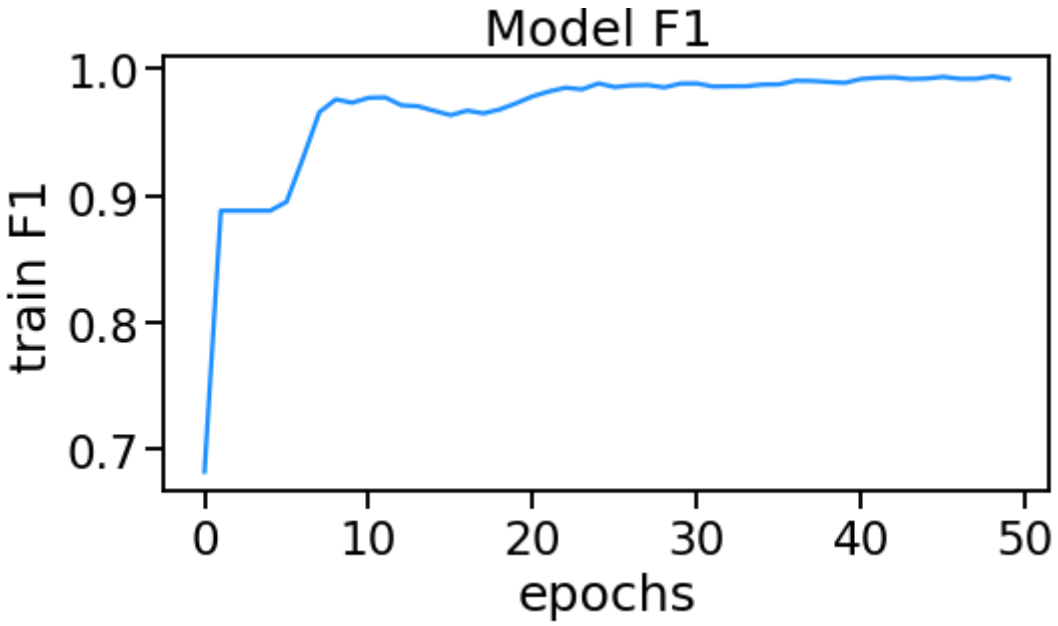
5.2.2 Federated learning


Model Convergence
train loss
0.6
0.4
0.2
0
10
20
30
40
50
epochs


Model Convergence
train accuracy
1.0
0.8
0.6
0
10
20
30
40
50
epochs


Model Precision
train precision
1.0
0.9
0.8
0
10
20
30
40
50
epochs


Model Recall
train recall
1.0
0.8
0.6
0
10
20
30
40
50
epochs



Test Results

- Loss: 0.08027677967523535
- Accuracy: 0.9609375
- Precision: 0.9579552790759688
- Recall: 0.9969135802469137
- F1-Score: 0.9767397542647266

[4] 5.3 Performance Comparisson

Although the performance of both the models is quiet similar, the comparison table below shows that covnext has relatively higher performance than Darknet. It is because pretrained weights of covnext on over 1 million images were used.

Model	FL or ML	Precision	Recall	F1	Accuracy	Loss
CNN (Scratch)	ML	96.94%	94.37%	95.63%	98.44%	0.0013046801645677075
	FL	89.86%	94.42%	91.38%	96.06%	0.10101205219204228
Diff %	**ML vs. FL**	7.08%	0.05%	4.25%	2.38%	0.0997073720274745725
CNN (Pretrained)	ML	97.42%	93.17%	95.05	98.44%	0.0013672651553023902
	FL	95.80%	99.70%	97.67%	96.01	0.08027677967523535
Diff %	**ML vs FL**	1.62%	6.53%	2.62%	2.43%	0.0789095145199329598
Improve % Pre-trained vs Scratch	**ML**	0.48%	1.2%	0.58%	0%	0.0000625849907346827
	FL	5.94%	5.26%	6.29%	0.03%	0.02073527251680693

[5]

Chapter 6: Conclusion

Following the emergence of COVID-19, extensive research has been conducted to develop methods for detecting it using Artificial Intelligence. Federated Learning is superior to standard Machine Learning in that it protects the privacy of users' data and consumes less computational power. As such, the purpose of this project is to construct a federated learning model capable of detecting COVID-19 in chest x-ray pictures. The method that will be taken is to conduct a comprehensive study on existing work on "COVID-19 detection using Federated Learning" and to construct a hybrid model by merging the most effective techniques. We have evaluated the research that has been conducted on "COVID-19 Detection Using Federated Learning". We started by constructing a machine learning model for COVID 19 detection in x-ray pictures. The model was created on Google Colab using the Python programming language and the PyTorch package. We federated the model once it converged by developing client and server classes in a single notebook. The results indicated that the FL model converged more slowly than the typical ML model.

For FYP-2, we begin by finishing the implementation of the FL model by generating distinct notebooks for the server and each client. Once a model was constructed and converged, we moved on to client-server orchestration. We implemented client-server Orchestration by creating a separate notebook for clients. Then we implemented server-side. Endpoints were created to support communication between client and server. Client-server orchestration was then tested thoroughly.

Finally, we intend to provide a GUI to show client-server interactions. Then, the server-side will be deployed on the Cloud. After the deployment, we will simulate the system's outcomes to see whether it is ready for implementation in a real-world environment.

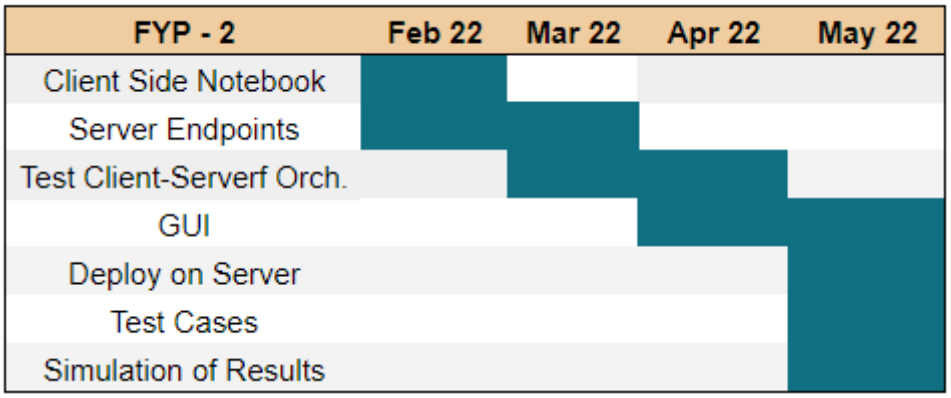


Figure 9: FYP-2 Work Plan
Plan for FYP-2

[6] References

[7] Q. Yang, Y. Liu, T. Chen, Y. Tong, "Federated Machine Learning: Concept and Applications", *ACM Transactions on Intelligent Systems and Technology* [Online]. vol. 10, issue 2, Jan 2019. Available: https://arxiv.org/abs/1902.04885v1. [Accessed: Aug. 12, 2021].

[8] Bhowal P, Sen S, Yoon JH, Geem ZW, Sarkar R., "Choquet Integral and Coalition Game-based Ensemble of Deep Learning Models for COVID-19 Screening from Chest X-ray Images," *IEEE J Biomed Health Inform*, Sep 2021. Available: https://pubmed.ncbi.nlm.nih.gov/34499608

[9] Javaheri, T., Homayounfar, M., Amoozgar, Z. et al."CovidCTNet: an open-source deep learning approach to diagnose covid-19 using small cohort of CT images," *npj Digit. Med* [Online]. vol. 4, issue 29, Feb 2021. Available: https://doi.org/10.1038/s41746-021-00399-3.

[10] Sanket, S., Vergin Raja Sarobin, M., Jani Anbarasi, L. et al. "Detection of novel coronavirus from chest X-rays using deep convolutional neural networks," *Multimed Tools Appl,* Sep 2021. Available: https://doi.org/10.1007/s11042-021-11257-5

[11] Ozturk T, Talo M, Yildirim EA, Baloglu UB, Yildirim O, Acharya UR, "Automated detection of COVID-19 cases using deep neural networks with X-ray images," *Computers in biology and medicine*, Jun 2020. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7187882/

[12] Jain, R., Gupta, M., Taneja, S. et al. "Deep learning based detection and analysis of COVID-19 on chest X-ray images," *Appl Intell*, March 2021. Available https://doi.org/10.1007/s10489-020-01902-1

[13] Khasawneh, N., Fraiwan, M., Fraiwan, L., Khassawneh, B., & Ibnian, A., "Detection of COVID-19 from Chest X-ray Images Using Deep Convolutional Neural Networks," *Sensors*, vol. 21, issue 5940, Sep 2021. Available: https://doi.org/10.3390/s21175940

[14] Makris, Antonios, Ioannis Kontopoulos, and Konstantinos Tserpes, 'COVID-19 Detection from Chest X-Ray Images Using Deep Learning and Convolutional Neural Networks,' *MedRxiv*, January 2020. Available: https://doi.org/10.1101/2020.05.22.20110817

[15] Harsh Panwar, P.K. Gupta, Mohammad Khubeb Siddiqui, Ruben Morales-Menendez, Vaishnavi Singh, "Application of deep learning for fast detection of COVID-19 in X-Rays using nCOVnet," *Chaos, Solitons & Fractals*, vol.138, issue 109944, May 2020. Available: https://pubmed.ncbi.nlm.nih.gov/32536759/

[16] Bassi, P.R.A.S., Attux, R., "A deep convolutional neural network for COVID-19 detection using chest X-rays," *Res. Biomed*, April 2021. Available: https://doi.org/10.1007/s42600-021-00132-9

[17] N. Narayan Das, N. Kumar, M. Kaur, V. Kumar, D. Singh, "Automated Deep Transfer Learning-Based Approach for Detection of COVID-19 Infection in Chest X-rays," *IRBM*, July 2020. Available: https://doi.org/10.1016/j.irbm.2020.07.001.

[18] Das, A.K., Ghosh, S., Thunder, S. et al. "Automatic COVID-19 detection from X-ray images using ensemble learning with convolutional neural network," *Pattern Anal Applic,* vol. 24, pp. 1111–1124, March 2021. Available: https://doi.org/10.1007/s10044-021-00970-4

[19] J. Zhang, L. Zhou, and L. Wang, "Triple-view Convolutional Neural Networks for COVID-19 Diagnosis with Chest X-ray," [Online]. Oct 2021. Available: https://arxiv.org/pdf/2010.14091.pdf [Accessed: Nov. 01, 2021]

[20] Tanvir Mahmud, Md Awsafur Rahman, Shaikh Anowarul Fattah, "CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature

optimization," *Computers in Biology and Medicine*, vol. 122, July 2020. Available: https://doi.org/10.1016/j.compbiomed.2020.103869.

[21] P. K. Sethy, S. K. Behera, P. K. Ratha, and P. Biswas, "Detection of coronavirus Disease (COVID-19) based on Deep Features and Support Vector Machine," *International Journal of Mathematical, Engineering and Management Sciences,* vol. 5, issue 4, pp. 643–651, 2020. Available: https://www.ijmems.in/volumes/volume5/number4/52-IJMEMS-20-46-54-643-651-2020.pdf

[22] Abdul Salam M, Taha S, Ramadan M, "COVID-19 detection using federated machine learning," *PLOS ONE*, June 2021. Available: https://doi.org/10.1371/journal.pone.0252573

[23] Trang-Thi Ho, Yennun-Huang, "DPCOVID: Privacy-Preserving Federated Covid-19 Detection," *arXiv*, Oct 2021. Available: https://arxiv.org/pdf/2110.13760.pdf

[24] Ines Feki, Sourour Ammar, Yousri Kessentini, Khan Muhammad, "Federated learning for COVID-19 screening from Chest X-ray images," *Applied Soft Computing*, vol 106, July 2021. Available: https://doi.org/10.1016/j.asoc.2021.107330.

[25] S. Banerjee, Misra, M. Prasad, E. Elmroth, and M. Bhuyan, "Multi-diseases Classification from Chest-X-ray: A Federated Deep Learning Approach," *AGCAI* [Online]. Available: https://opus.lib.uts.edu.au/bitstream/10453/148730/2/Accepted-AJCAI-paper_29.pdf [Accessed: Nov. 01, 2021]