Solo Project: Federated Learning

Jizheng Chen 520021911182

April 18, 2023

1 Introduction

Federated Learning can be described as an on-device, collaborative ML setting, which was introduced by Google in 2015[1]. It's training process can be described as Figure 1[2], a global ML model that lives on a central service provider is shared across a pool of clients (or users) by being broadcasted to their local edge devices. The global model is used as an initial model and trained on each client's data on the local device, resulting in a trained local model and client data remaining on the client edge device. Once a local model is trained, the updated parameters of the client are sent back to the central server in order to be aggregated across the pool of participating clients and used to update the global model parameters. Once the global model is updated, a training round of FL is complete, and the updated global model is broadcasted to another selected pool of clients to run a second round of training. FL is an iterative and collaborative learning setting. Due to the modularized nature of FL, it fosters an easy environment for private and secure collaborative ML. It is typically at the step of aggregating the client local parameter updates where several secure computations can be incorporated to enforce the privacy and security of client data and the global model trained, which, in general, can be described as:

- **Model Initialization**: An initial model was created by central server
- **Model Transmission**: The model is issued to clients for their separate training.
- **Local Training**: Each client separately train their models using isolated data.
- **Update Transmission**: Clients send their trained models to the server with updated parameters.
- **Global Training**: The central server use a certain way to aggregate model parameters from models trained by Client ends.

In this project, the process in real application scenarios was simplified, where, we use file reading and writing or socket communication to simulate the procedure of model transmission between server end and client ends. In this setting,

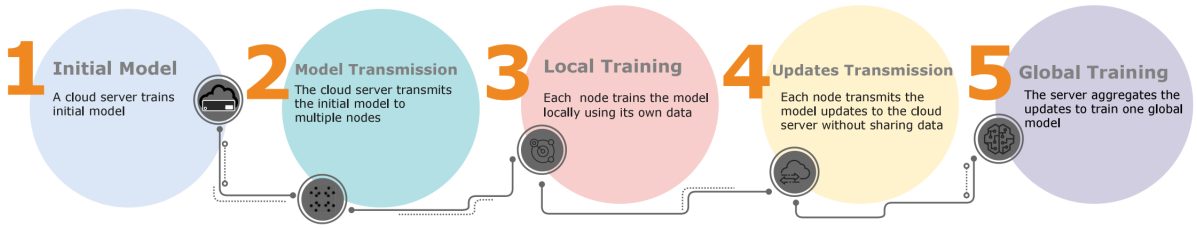

Figure 1: Federated Learning Process

we suppose that one server and N clients train a classification task on MNIST dataset collaboratively, which updates model parameters by **a) direct data access or b) ".pth" file reading and writing.**

We have three stages to simulate the process of federated learning: Stage 1 uses a basic training process of federated learning, in which the server first delivers the global model to all clients, than N clients each trains the local model on its private data. Finally, a average aggregating process is applied by the server to integrate the global model from the local models of all clients, and the global model is verified on the test dataset of MNIST.

Stage 2 lets M out of N clients be selected to participate in the update in each round. It uses a partial participation mode to randomly select a subset of clients to deliver the initial model and receive the updated model parameters from those selected models.

Compare with stage 1 and stage 2's interaction between server and client by model file reading and writing, here the server and client use **Socket Communication** to interact with each other. Specifically a TCP socket is established by the server and it keeps listening until a certain client gets connected to it. Then the server handout the initial model to this client, receives its updated parameter and does a gradual model aggregation process. The process lasts until all the clients get updated and trained, and the server thus has a new model for next epoch.

Above is the general explanation of the three stages, next sections will be illustrated as follows: section 2 gives the experimental setting and details of this FL project, and section 3 experiment results is elaborated. Finally in section 4 I will give a brief conclusion.[1]

2 Experiment Settings

In my simulation experiments, the workflow of every training epoch can be illustrated at Figure 3:

[1] All the code and results is available at https://github.com/Otsuts/TCP-socket-based-FL-

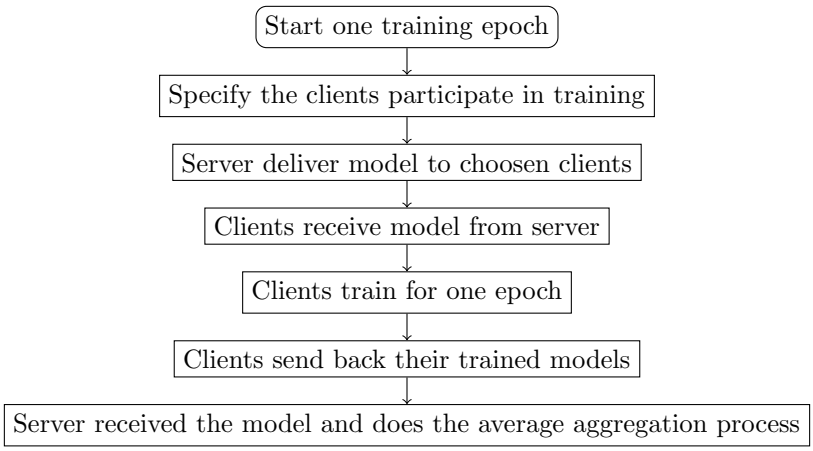

Figure 2: Workflow for FL simulation

2.1 Stage 1

Stage 1 deals with the workflow of federated learning process as follows:

- **Specify the clients participate in training**:In this step all the clients take part in the update process, i.e. no more extra operations are down to distinguish the clients.
- **Server deliver model**: in the model file reading and writing setting, this step can be simply realized by storing the initial model's '.pth' file in a public file path that all the clients can read.
- **Clients receive model from server**: in file reading setting, this step is also easily realized by letting the clients read and load the parameters from the public path.
- **Clients train for one epoch**: each client train their model for one epoch using their own private data.
- **Clients send back their trained models**: This can be realized by each clients separately stores their model in a '.pth' file that the server has access to.
- **Server aggregate model**: in this step the server read the model parameters from clients' file path and average the model parameters to get a new model for further training epochs.

In my experiment process, each clients' learning rate is set to 1e-3, with Adam optimizer, and the training epoch is set to 30, which according to experimental results, has reached a convergence point.

2.2 Stage 2

Stage 2 is somewhat like stage 1 in model transmission process, except that M out of N clients are selected to participate in the update in each round. The learning process is as follows:

- **Specify the clients participate in training**:In this step a random subset of clients are generated according to the number of selected clients M, and their references are transferred to next stages.
- **Server deliver model**: in the model file reading and writing setting, this step can be realized by storing the initial model's '.pth' file in a public file path that the selected subset of clients can read.
- **Clients receive model from server**: in file reading setting, this step is also realized by letting the selected set of clients read and load the parameters from the public path.
- **Clients train for one epoch**: the participated clients train their model for one epoch using their own private data.
- **Clients send back their trained models**: This can be realized by letting participated clients stores their model in a '.pth' file that the server has access to, and the server model can only do aggregation from those newly stored models.
- **Server aggregate model**: in this step the server read the model parameters from clients' file path and average the model parameters to get a new model for further training epochs.

In stage 2 the implementation details are similar to that in stage 1, except that in the learning process when participated client number is small, the server model performance is greatly damaged, with is corresponding to the findings, and more details will be experimented and demonstrated in section 3

2.3 Stage 3

In section 3 things gets more interesting. The usage of socket communication simulates the real training scenarios more, and more difference lies in:

1. Socket communication makes interaction between clients and server more subjective and flexible, that means, clients can individually choose the timestamp to connect with the server, and when there are no transmission, server kept listening.
2. During the process of training the clients comes one by one, and the server should listen on the port, accept local models from each client and gradually complete model aggregation.

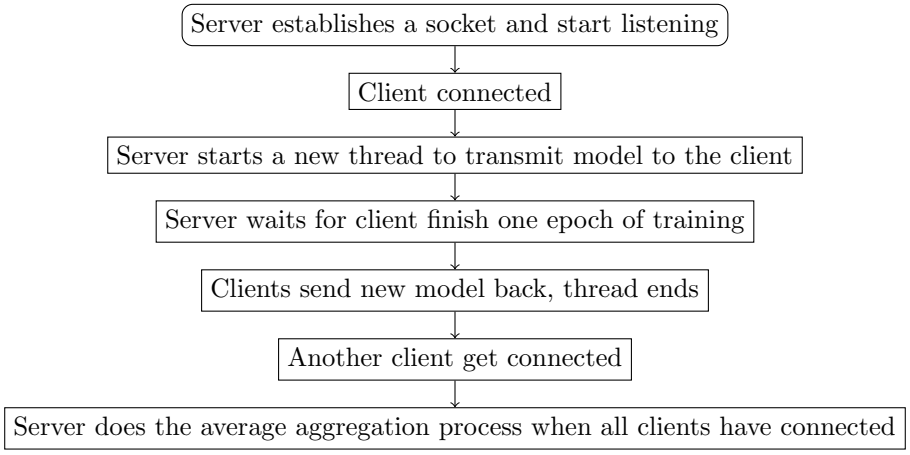

Figure 3: Workflow for TCP training for one epoch

So in stage 3, the work flow of training is as follows, which, in my case, is slightly different from the above two stages:

To deal with model transmission in TCP protocol, I used the **pickle** module to convert the model parameter dict from received string, and due to the reason that TCP communication has max capacity in one single package, the data must by transmitted through many packages and gets concatenated to become the complete model.

3 Experiment Results

3.1 Results in stage 1 stage 2

The following figure 4 shows the relationship between the number of participants and the accuracy, where, when client number becomes 20, it is the case in stage 1 that every client participates in the updating process.

From the above figure several conclusions can be drawn:

1. The best accuracy in stage 1 and stage 2 are both close to 100%, prooving that our implementation of federal leaning is correct and successful, realizing a good result.

2. When the number of clients participated is small, the performance of our model is greatly damaged, and the training process can't converge. That is the case when M is 5 and 10.

3. When training client becomes large enough, the model can converge, and when M is larger, the performance is better, which is consistent with our intuition.

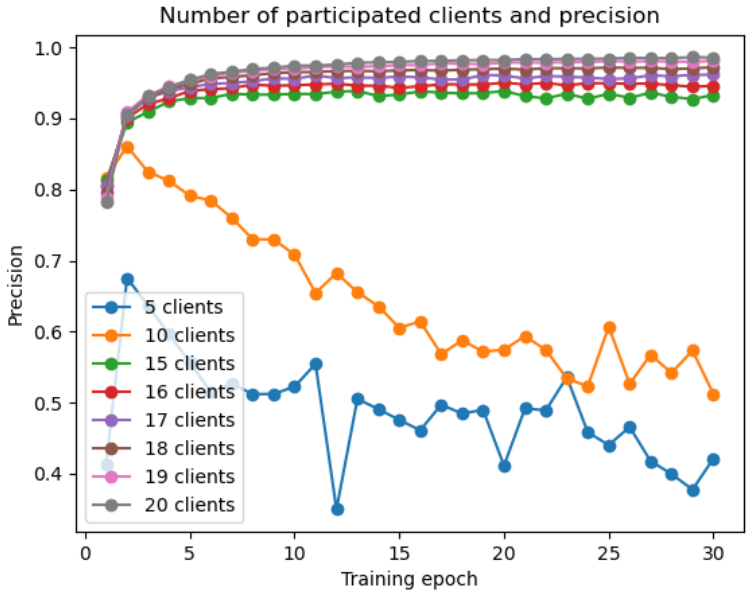

Figure 4: Relationship between participated clients and precision

3.2 Stage 3

The result of stage 1 and stage 3 is at Figure 5, where similar converged result is gotten, showing the correctness of socket communication way of model transmission.

The following tabular gives a statistical results of the three stages' experimental results.

Number of Clients participated	**Best acc**
5	67.52
10	86.09
15	93.62
16	95.02
17	96.16
18	97.21
19	98.12
20	98.64
20(stage3)	98.32

4 Conclusion

In this federated learning project, I initially tried the basic paradigm of federated learning and achieved good results. At the same time, I also realized

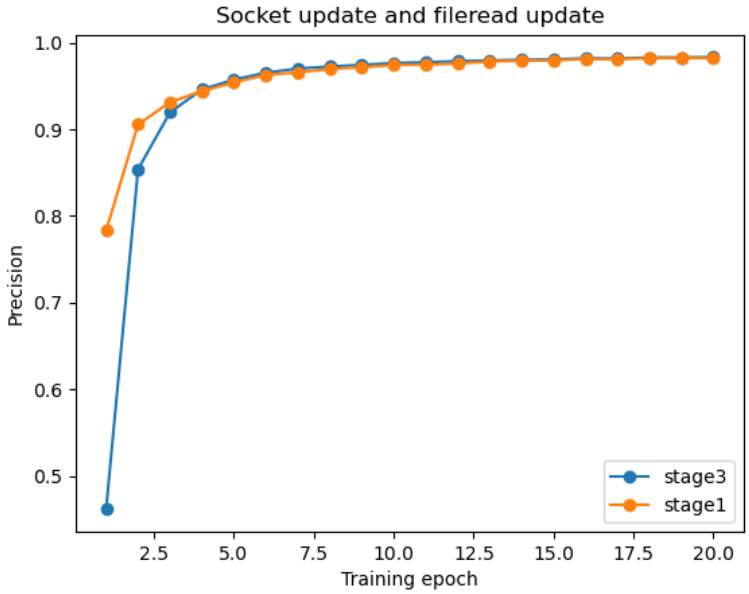

Figure 5: Stage 1 and stage3

that compared with general machine learning, federated learning faces the challenges of convergence is difficult to guarantee, message transmission consumes large resources, and there are various heterogeneous challenges.

References

[1] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[2] Syreen Banabilah, Moayad Aloqaily, Eitaa Alsayed, Nida Malik, and Yaser Jararweh. Federated learning review: Fundamentals, enabling technologies, and future applications. *Information processing & management*, 59(6):103061, 2022.