DistLearn: Federated Learning in Distributed Systems

Abhinav Raundhal
2022101089
IIIT Hyderabad

Archisha Panda
2022111019
IIIT Hyderabad

Vinit Mehta
2022111001
IIIT Hyderabad

I. Introduction

Federated Learning (FL) is a distributed machine learning approach where multiple clients collaboratively train a global model without sharing raw data. However, FL faces challenges such as privacy risks, where model updates may still leak sensitive information, security threats, including Byzantine attacks and model poisoning and communication overhead as frequent model updates can strain network bandwidth. Despite these challenges, FL offers significant advantages like preserving data privacy by keeping data local, reducing data transfer costs, enabling collaborative learning across organizations without exposing confidential data, and supporting model personalization for individual devices. Overcoming these challenges is key to realizing the full potential of privacy-preserving, decentralized AI. Our goal is to develop a Distributed Federated Learning (DFL) system that overcomes selected few of these challenges and enables scalable, secure, and communication-efficient FL training.

II. System Architecture

Our DFL framework consists of the following components:

- **Central Server:** The central server stores and manages the global model and aggregates local model updates to ensure continuous improvement.
- **Clients:** Each client has its own private dataset which never leaves the device. Clients could be mobile devices, edge devices, or enterprise servers participating in federated learning.
- **Model Distribution:** The central server sends an initial copy of the global model to all selected clients which serves as a starting point for local training at each client.
- **Local Training:** Each client trains the model independently using its own dataset and updates the model weights based on local optimization. Techniques like gradient descent, adaptive learning rates, and momentum-based optimization can be used to improve training.
- **Model Update Transmission:** After local training, each client encrypts and transmits the updated model weights to the central server.
- **Global Aggregation:** The central server collects the updated weights from all clients and uses an aggregation strategy (e.g., Federated Averaging (FedAvg), adaptive weighting) to merge the updates into the global model. Aggregation techniques ensure that relevant updates are given more weight to improve model accuracy.
- **Model Convergence:** The updated global model is redistributed to clients for the next round of training. This iterative process continues until both local and global models converge to an optimal performance level.

III. Baseline Model

Our baseline model is based on a research paper by Google, where they introduced the **FedSGD** algorithm for distributed learning which includes the following steps:

1) The server defines a basic model architecture, initializes it with random weights, and distributes a copy of the model to each client.
2) Each client performs a single forward pass using its own dataset. The entire dataset is passed through the model, and the final gradient is computed as the average of gradients obtained from each data point's forward pass. The client then sends only these averaged gradients back to the central server.
3) The central server aggregates the gradients received from all clients, weighting them based on the number of data points each client possesses. Specifically, if client k has n_k data points, then the weight of its gradients is given by $\frac{n_k}{n}$, where $n = \sum_{i=0}^{N} n_i$ and N is the total number of clients. The server then updates the global model using these weighted averaged gradients.
4) The updated global model is sent back to the clients, and this process repeats until the model achieves a satisfactory performance level.

This method allows the model to improve iteratively while ensuring that clients' private data remains secure and never leaves their devices.

IV. Challenges and Proposed Solutions

For the scope of this project, we will be focusing on the following challenges and build solutions to solve them.

A. Communication Efficiency

Frequent transmission of model updates after every epoch leads to significant communication costs, sometimes exceeding the computational overhead. To optimize communication efficiency, we propose the following strategies:

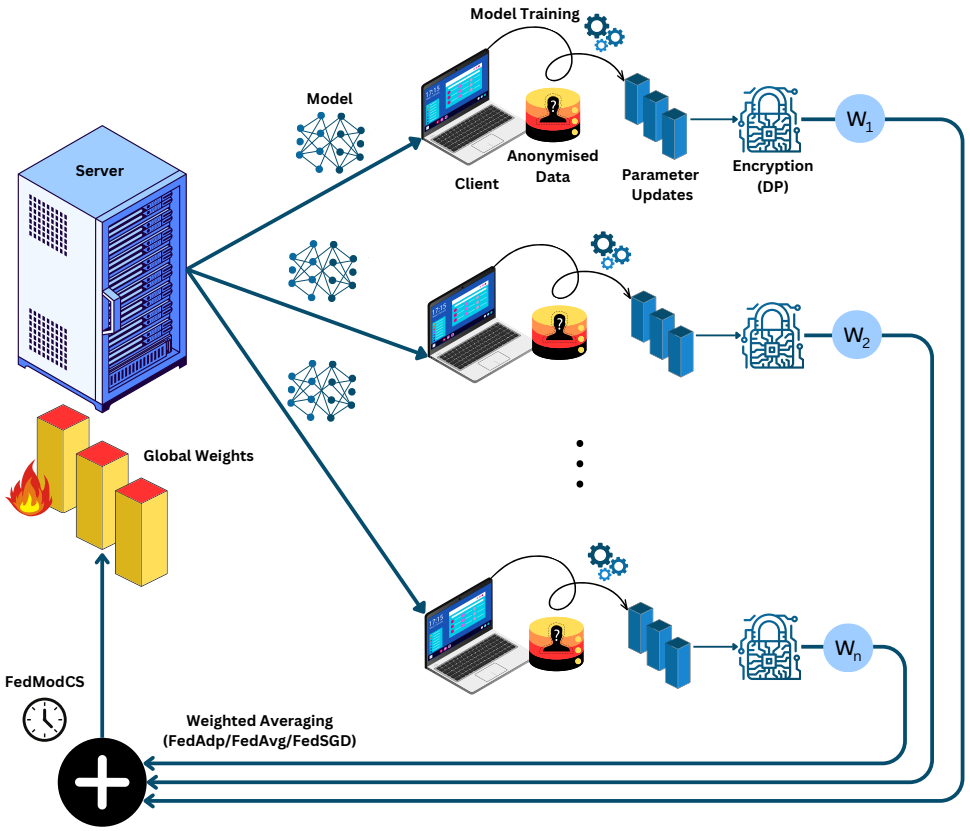

Fig. 1. Federated Learning Architecture: The figure illustrates a central server that coordinates training across multiple clients without accessing their local data. The server distributes an initial global model to selected clients, which train the model using their local datasets. The locally updated parameter weights are then encrypted to ensure security before being sent back to the server. The server aggregates these encrypted updates using a weighted federated averaging mechanism to update the global model weights. The process iterates until convergence, ensuring collaborative learning while preserving data privacy across distributed clients.

- **Federated Averaging (FedAvg):** Instead of sending updates after every local iteration, clients perform multiple local training epochs before transmitting model updates to the central server.
- **Adaptive Federated Averaging: (FedAdp):** For this variation, we will focus on assigning adaptive weights to the clients instead of randomly selecting them, as done in FedAvg. This is based on cosine similarity between the local gradient vector and global gradient vector.
- **Client Selection (based on FedCS):** Since not all clients contribute equally to the global model due to differences in computational power, network stability, and data quality, an effective client selection strategy ensures better model performance with minimal communication overhead. For the scope of this project, we focus on a subset of the FedCS algorithm, let's call it modified FedCS or FedModCS. We would be selecting the clients in by a two step process. First, candidate clients are selected on the basis of the size of data resources relevant to the current training task. The global weight updates only incorporate the weight updates from those clients which are able to locally train the model within a predefined deadline.

B. Privacy Protection in Federated Learning

When clients send updated model weights, an attacker could intercept and reverse-engineer them to infer sensitive data. To prevent this, we propose to implement the following method:

- **Differential Privacy (DP)**: Adds controlled noise to weight updates before transmission, making it difficult to extract original data. While this improves privacy, it can slightly reduce model performance as updates are less precise. We will compare the performance of federated models with and without DP.
- **Data Anonymization**: Clients remove personally identifiable information from local datasets before training by encoding (by some hashing methods) them as anonymous labels that ensures that even if some data is leaked, it cannot be linked to individuals.

V. Tech Stack

We will use the following technologies:

- **Programming Language:** Python
- **Frameworks:** PyTorch for deep learning
- **Communication:** gRPC for efficient communication

VI. Datasets and Models

We will evaluate our framework and compare the different approaches of Binary and Multi-class Classification using MLP and CNN architectures on the following datasets:

- **MNIST** It is a widely used benchmark dataset for handwritten digit classification consisting of 28x28 grayscale images.
- **Fashion MNIST** It is a dataset of 28x28 grayscale images of 10 fashion categories.
- **Diabetes Dataset** This dataset represents a real-world medical application for detecting diabetes based on other medical parameters.

VII. Timeline

- **Week 1:** Implementing baseline model FedSGD and FedAvg
- **Weeks 2 and 3:** Modify FedAvg by adding Client Selection (FedAdp and FedModCS)
- **Week 4:** Implement Differential Privacy and documenting the comparison of different approaches.

VIII. References

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR.

Wu H, Wang P (2021) Fast-Convergent Federated Learning With Adaptive Weighting. IEEE Trans Cogn Commun Netw 7(4):1078–1088

Nishio T, Yonetani R (2019) Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. In: Proceedings of ICC 2019–2019 IEEE International Conference on Communications (ICC) pp. 1–7