DistLearn

Distributed Federated Learning

Abhinav Raundhal (2022101089)
Archisha Panda (2022111019)
Vinit Mehta (2022111001)

Introduction

What is Federated Learning?

It all started with ...

Published in: PMLR

Year of Publish: 2017

Link: https://arxiv.org/pdf/1602.05629

Authors: H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Aguera y Arcas (Google)

Federated Learning (FL) enables multiple clients to collaboratively train a global model without sharing raw data, preserving privacy and reducing data transfer costs. However, FL faces challenges like privacy risks, security threats, and high communication overhead. Our project aims to build a Distributed Federated Learning (DFL) system that addresses these issues to achieve scalable, secure, and communication-efficient training.

Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan **Eider Moore** **Daniel Ramage** **Seth Hampson** **Blaise Agüera y Arcas**
Google, Inc., 651 N 34th St., Seattle, WA 98103 USA

Abstract

Modern mobile devices have access to a wealth of data suitable for learning models, which in turn can greatly improve the user experience on the device. For example, language models can improve speech recognition and text entry, and image models can automatically select good photos. However, this rich data is often privacy sensitive, large in quantity, or both, which may preclude logging to the data center and training there using conventional approaches. We advocate an alternative that leaves the training data distributed on the mobile devices, and learns a shared model by aggregating locally-computed updates. We term this decentralized approach *Federated Learning*.

We present a practical method for the federated learning of deep networks based on iterative model averaging, and conduct an extensive empirical evaluation, considering five different model architectures and four datasets. These experiments demonstrate the approach is robust to the unbalanced and non-IID data distributions that are a defining characteristic of this setting. Communication costs are the principal constraint, and we show a reduction in required communication rounds by 10–100× as compared to synchronized stochastic gradient descent.

1 Introduction

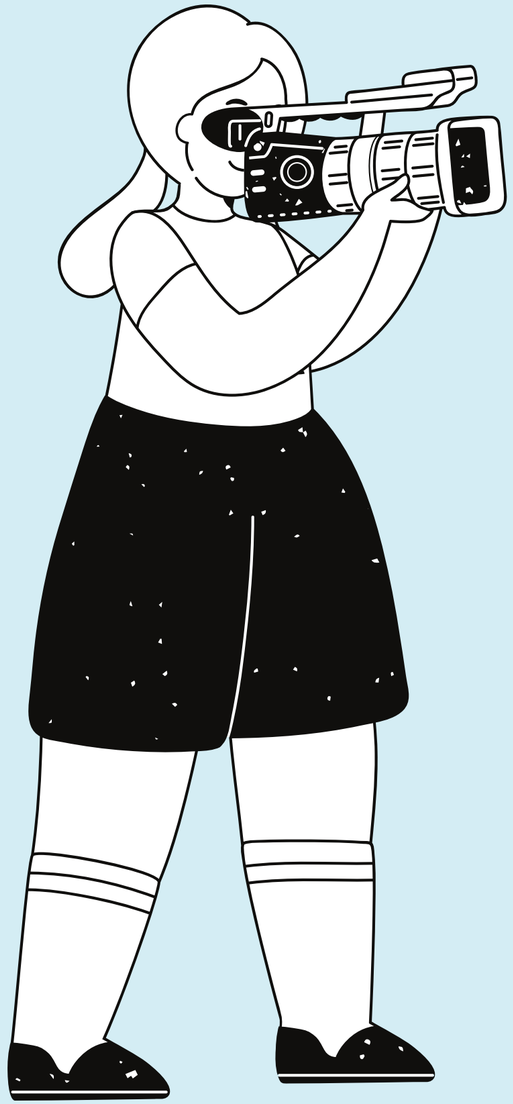
Increasingly, phones and tablets are the primary computing devices for many people [30, 2]. The powerful sensors on these devices (including cameras, microphones, and GPS), combined with the fact they are frequently carried, means they have access to an unprecedented amount of data, much of it private in nature. Models learned on such data hold the promise of greatly improving usability by powering more intelligent applications, but the sensitive nature of the data means there are risks and responsibilities to storing it in a centralized location.

We investigate a learning technique that allows users to collectively reap the benefits of shared models trained from this rich data, without the need to centrally store it. We term our approach *Federated Learning*, since the learning task is solved by a loose federation of participating devices (which we refer to as *clients*) which are coordinated by a central *server*. Each client has a local training dataset which is never uploaded to the server. Instead, each client computes an update to the current global model maintained by the server, and only this update is communicated. This is a direct application of the principle of *focused collection* or *data minimization* proposed by the 2012 White House report on privacy of consumer data [39]. Since these updates are specific to improving the current model, there is no reason to store them once they have been applied.

A principal advantage of this approach is the decoupling of model training from the need for direct access to the raw training data. Clearly, some trust of the server coordinating the training is still required. However, for applications where the training objective can be specified on the basis of data available on each client, federated learning can significantly reduce privacy and security risks by limiting the attack surface to only the device, rather than the device and the cloud.

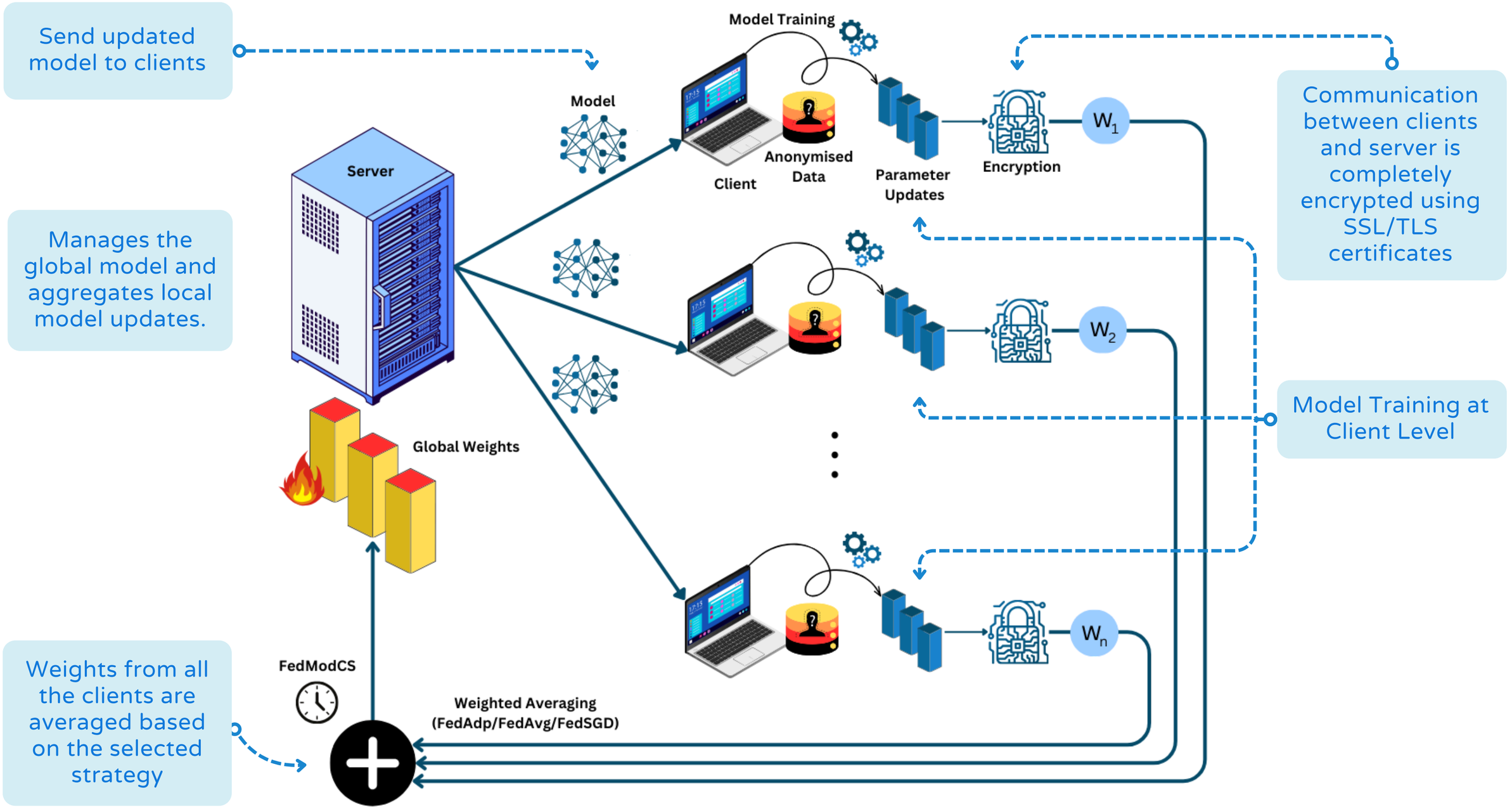
Our primary contributions are 1) the identification of the problem of training on decentralized data from mobile devices as an important research direction; 2) the selection of a straightforward and practical algorithm that can be applied to this setting; and 3) an extensive empirical evaluation of the proposed approach. More concretely, we introduce the `FederatedAveraging` algorithm, which combines local stochastic gradient descent (SGD) on each client with a server that performs model averaging. We perform extensive experiments on this algorithm, demonstrating it is robust to unbalanced and non-IID data distributions, and can reduce the rounds of communication needed to train a deep network on decentralized data by orders of magnitude.

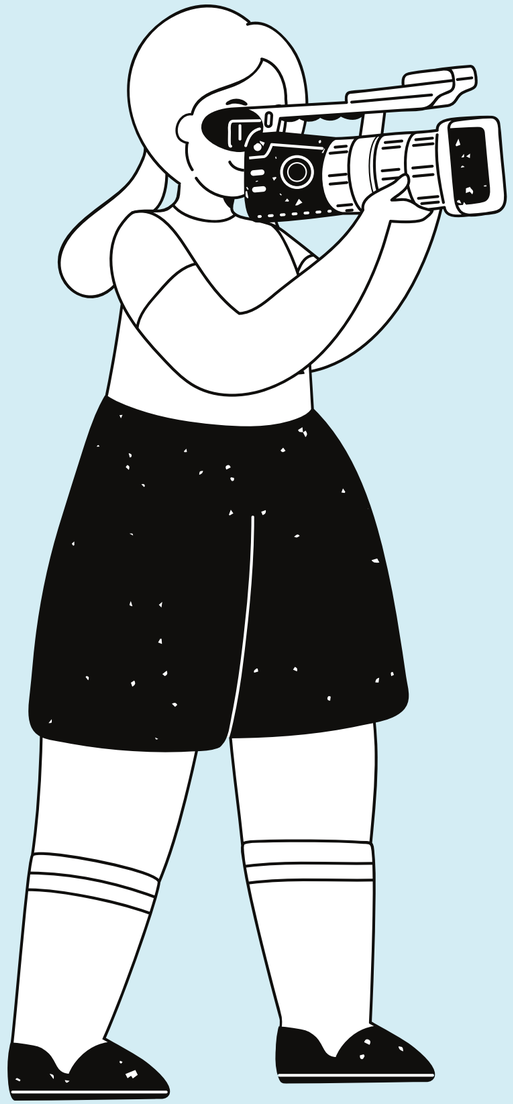
Appearing in Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Flordia, USA. JMLR: W&CP volume 54.



System Architecture

How clients and server interact to train the global model.


Send updated model to clients
Model Training
Model
Server
Client
Anonymised Data
Parameter Updates
Encryption
W_1
W_2
W_n
Communication between clients and server is completely encrypted using SSL/TLS certificates
Manages the global model and aggregates local model updates.
Global Weights
Model Training at Client Level
FedModCS
Weighted Averaging (FedAdp/FedAvg/FedSGD)
Weights from all the clients are averaged based on the selected strategy



Tech Used

Different technologies used for implementation

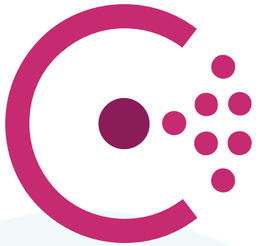
Frameworks Used

01

gRPC

Low-latency, scalable communication between server and clients using remote procedure calls

02



Consul

Dynamic discovery of active clients and services with fault tolerance and automatic registration

03

PyTorch

Flexible deep learning framework with GPU acceleration and support for model building and training

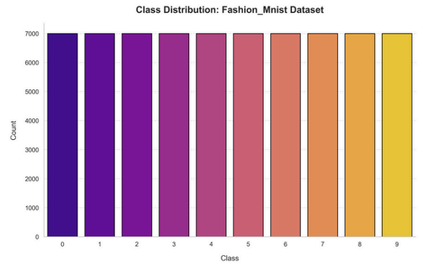
04

Open-SSL

Generates self-signed certificates and private keys for securing gRPC communication via TLS

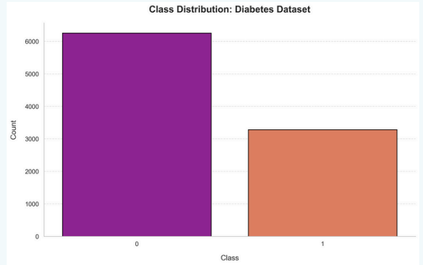
Datasets and Models

- **FashionMNISTCNN**
  - **CNN** with Conv + Pool + FC layers
  - Trained on 28x28 grayscale images (fashion categories)



- **MNISTMLP**
  - Benchmark dataset for handwritten digit classification
  - Consisting of 28x28 grayscale images
  - **MLP** with 3 FC layers

- **DiabetesMLP**
  - Tabular data with 16 features
  - **MLP** with 3 FC layers



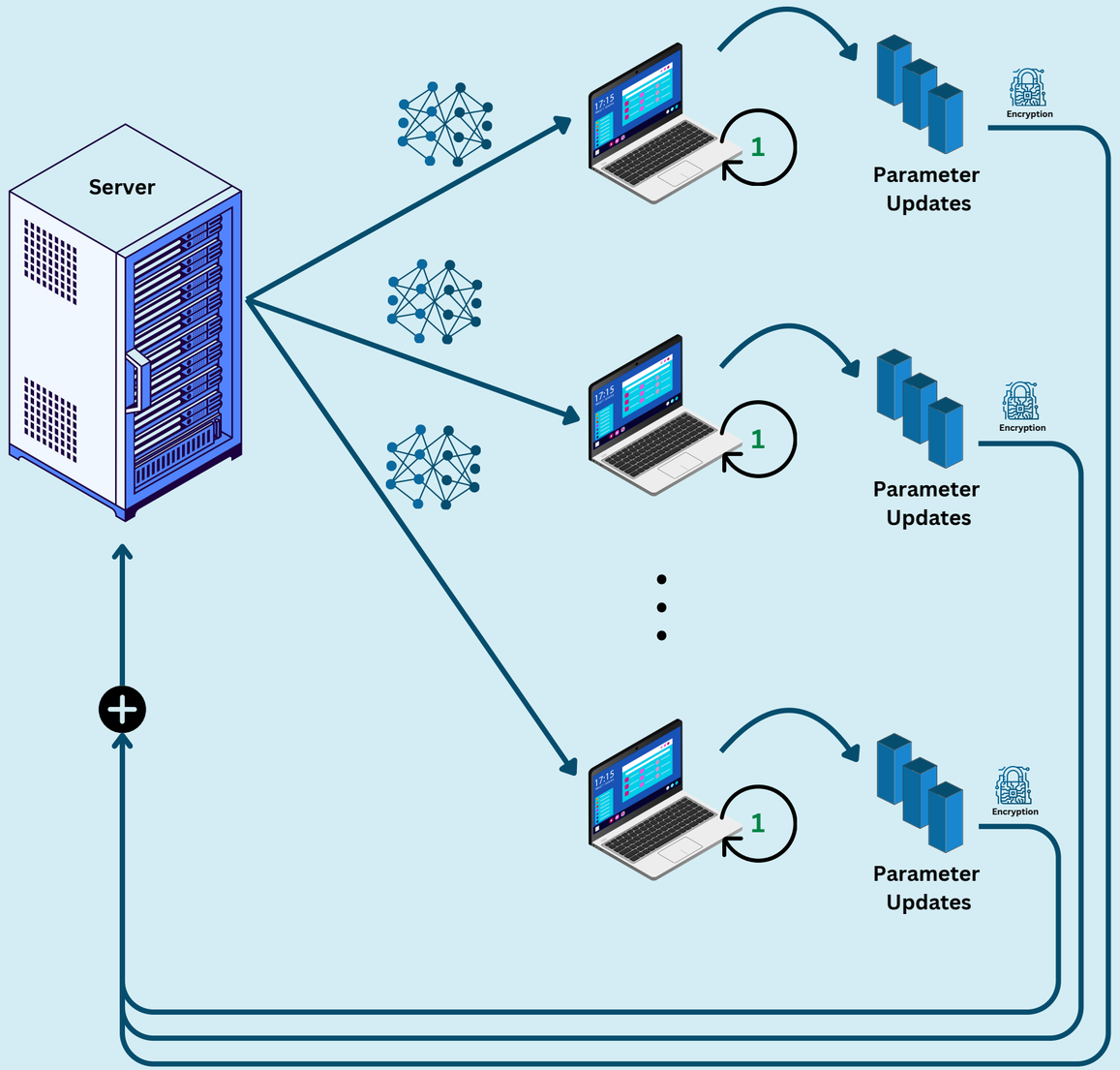
FedSGD: Baseline Model

Algorithm

Algorithm 1 FEDSGD. The K clients are indexed by k; B is the local minibatch size, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{CLIENTUPDATE}(k, w_t)$
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for

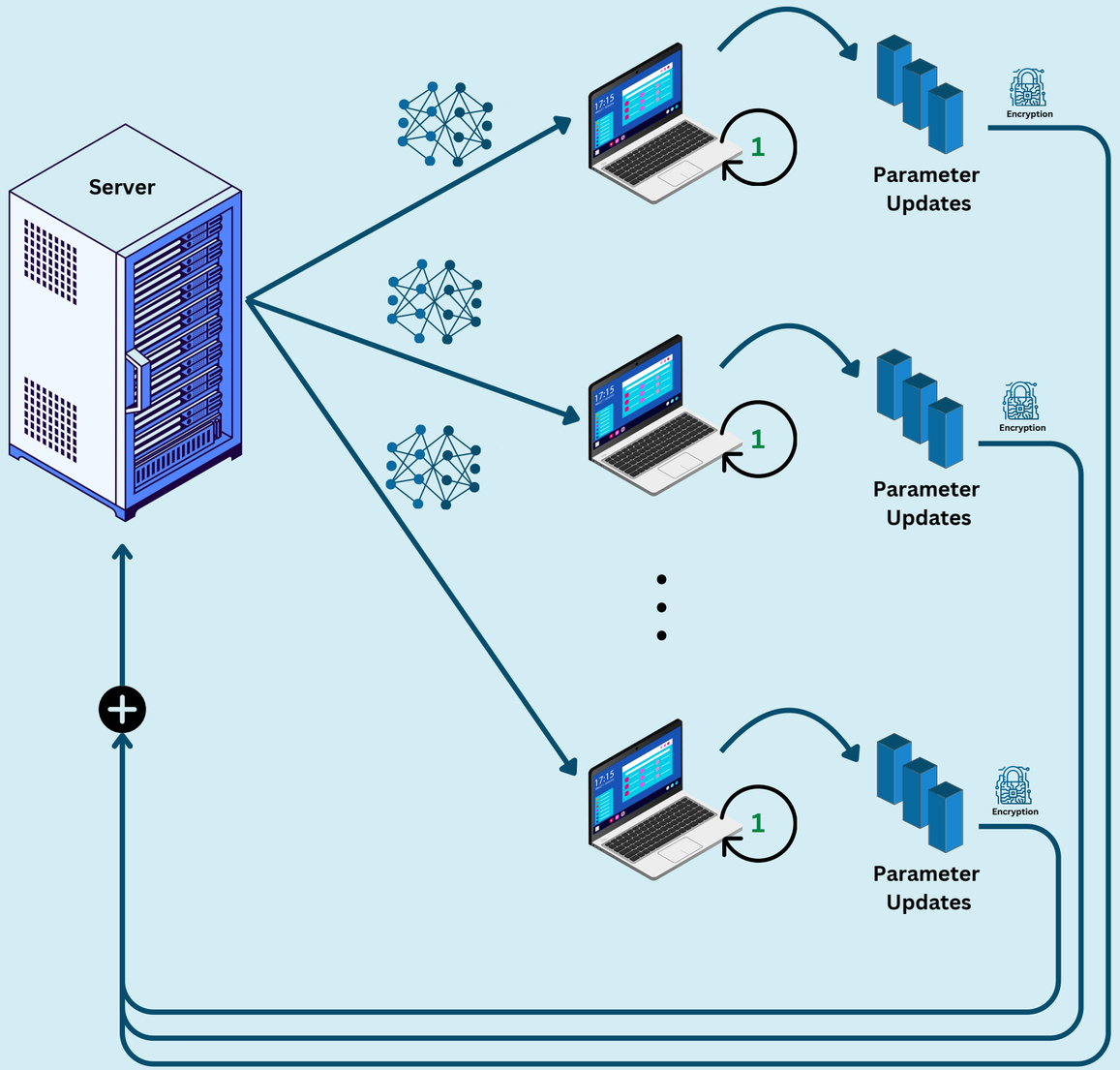
function CLIENTUPDATE(k, w): ▷ Run on client k
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
  end for
  return w to server
end function



Can we do better?

Problems to tackle

- Communication overhead
- Clients send updates after *every epoch*



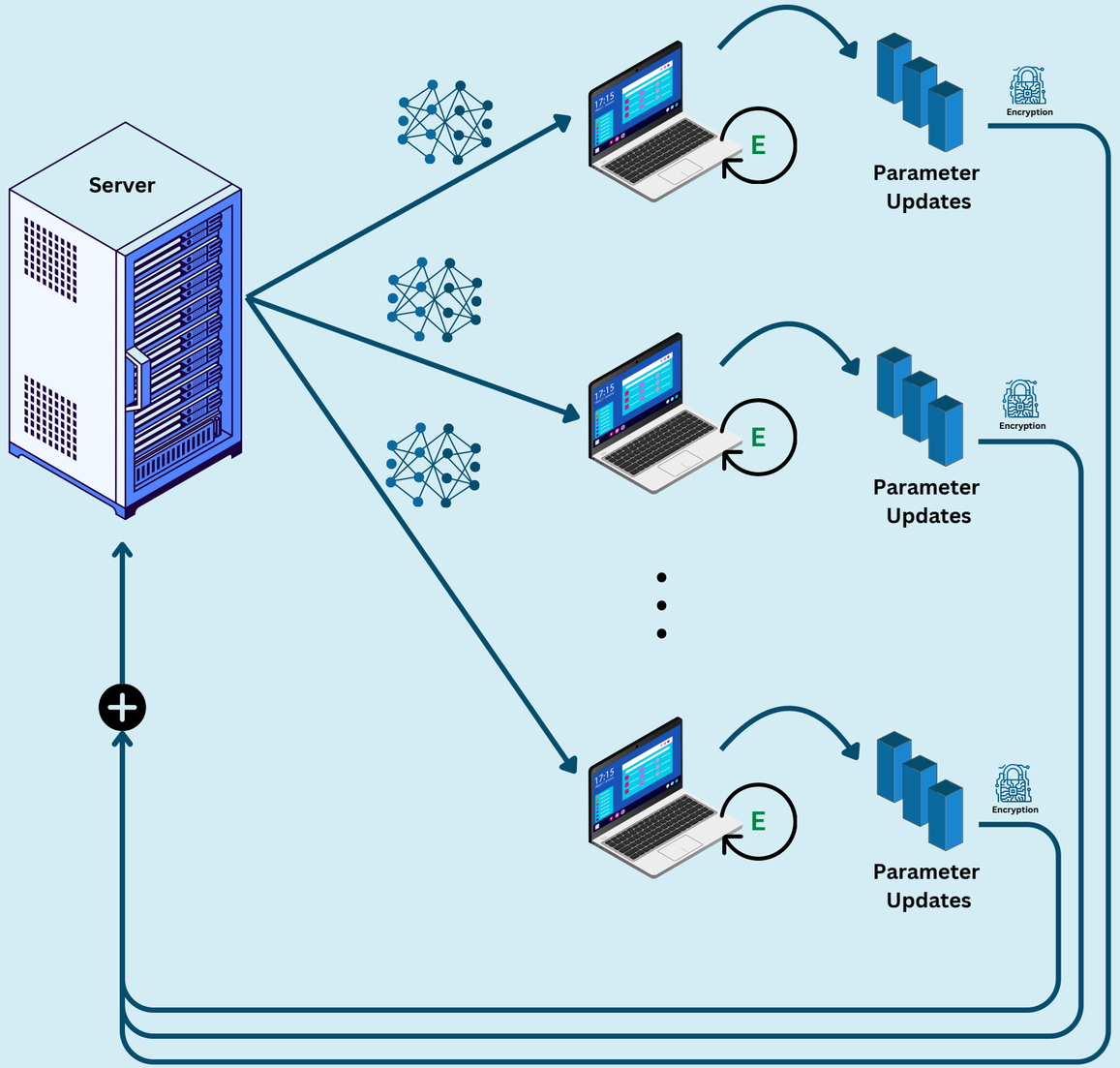
FedAvg: Enhanced Model

Algorithm

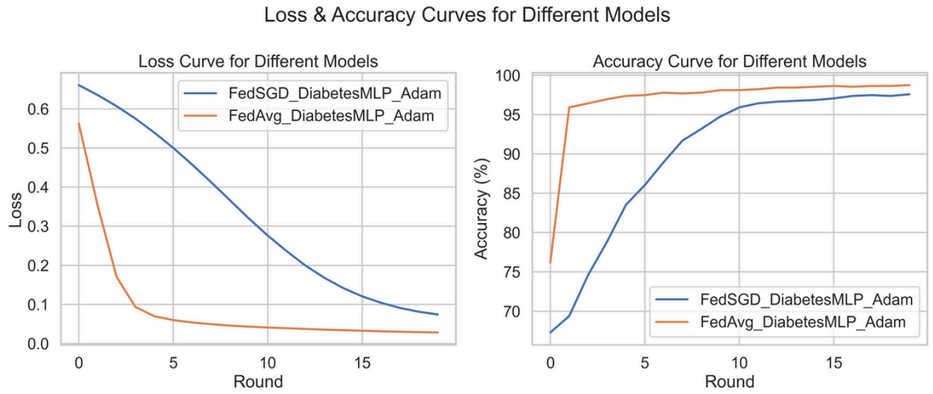
Algorithm 2 FedAvg. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for

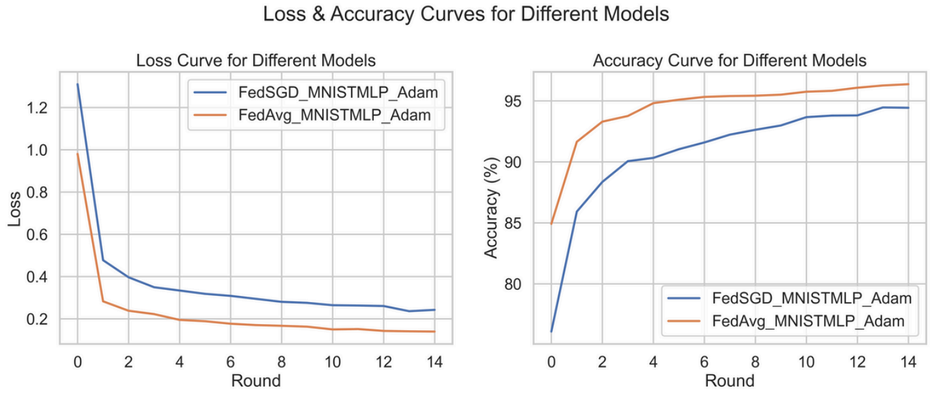
function ClientUpdate(k, w) ▷ Run on client k
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
    end for
  end for
  return w to server
end function



FedSGD vs FedAvg (Accuracy)

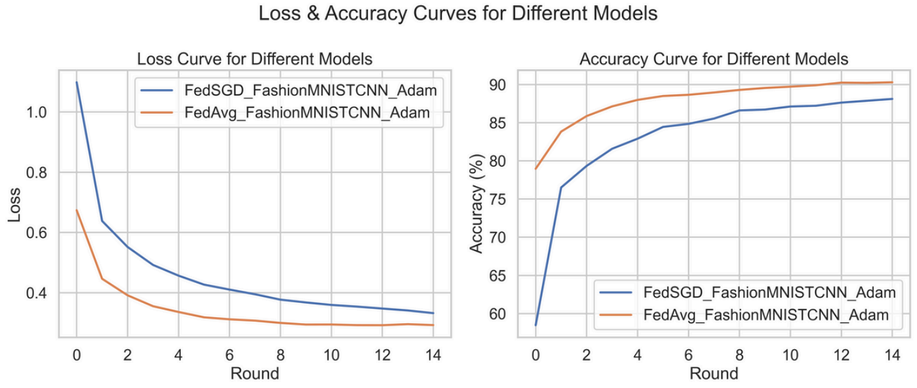


Diabetes MLP



MNIST MLP

In case of binary classification task (simpler than multi-class classification), the difference in model performance increases multi-fold



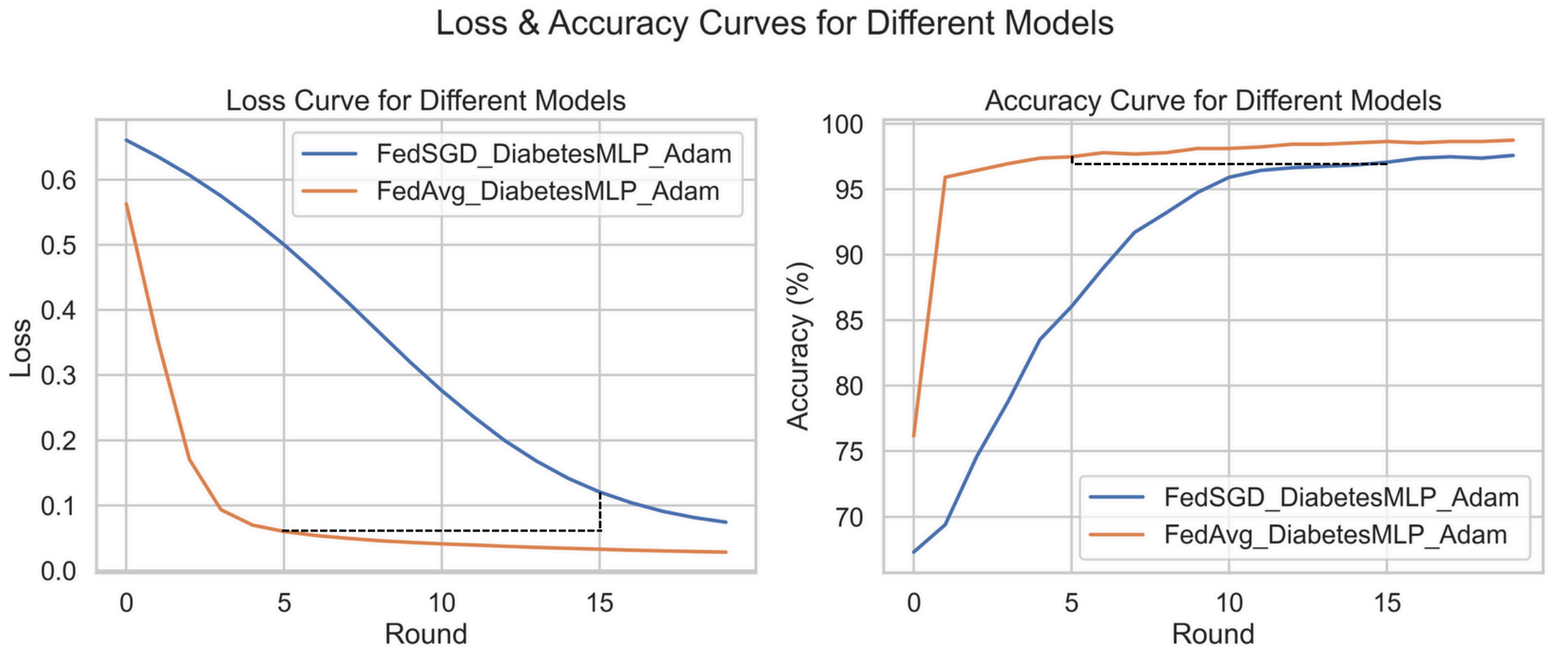
FashionMNIST CNN

FedAvg takes much less communication rounds than FedSGD to reach a given certain accuracy level. Thus reducing the communication overhead.

Note: Even if we compare the number of overall internal rounds (epochs), then also FedAvg performs better than FedSGD.

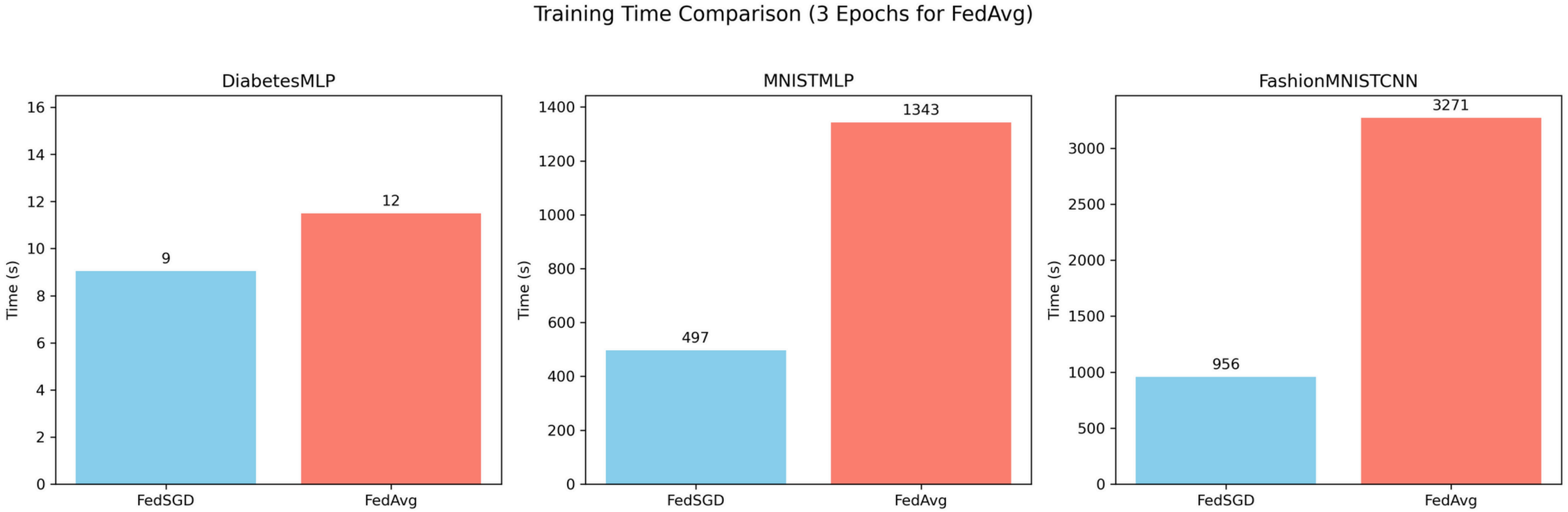
FedSGD vs FedAvg (Accuracy)

Note: Even if we compare the overall number of internal rounds (epochs) instead of the communication rounds, then also FedAvg performs better than FedSGD.
What FedAvg achieves at the end of 5 communication rounds (total 15 internal rounds) is better than what FedSGD achieves at the end of 15 communication rounds.



Diabetes MLP (FedAvg with 3 internal epohs)

FedSGD vs FedAvg (Time)



Time taken FedSGD vs FedAvg on all 3 models

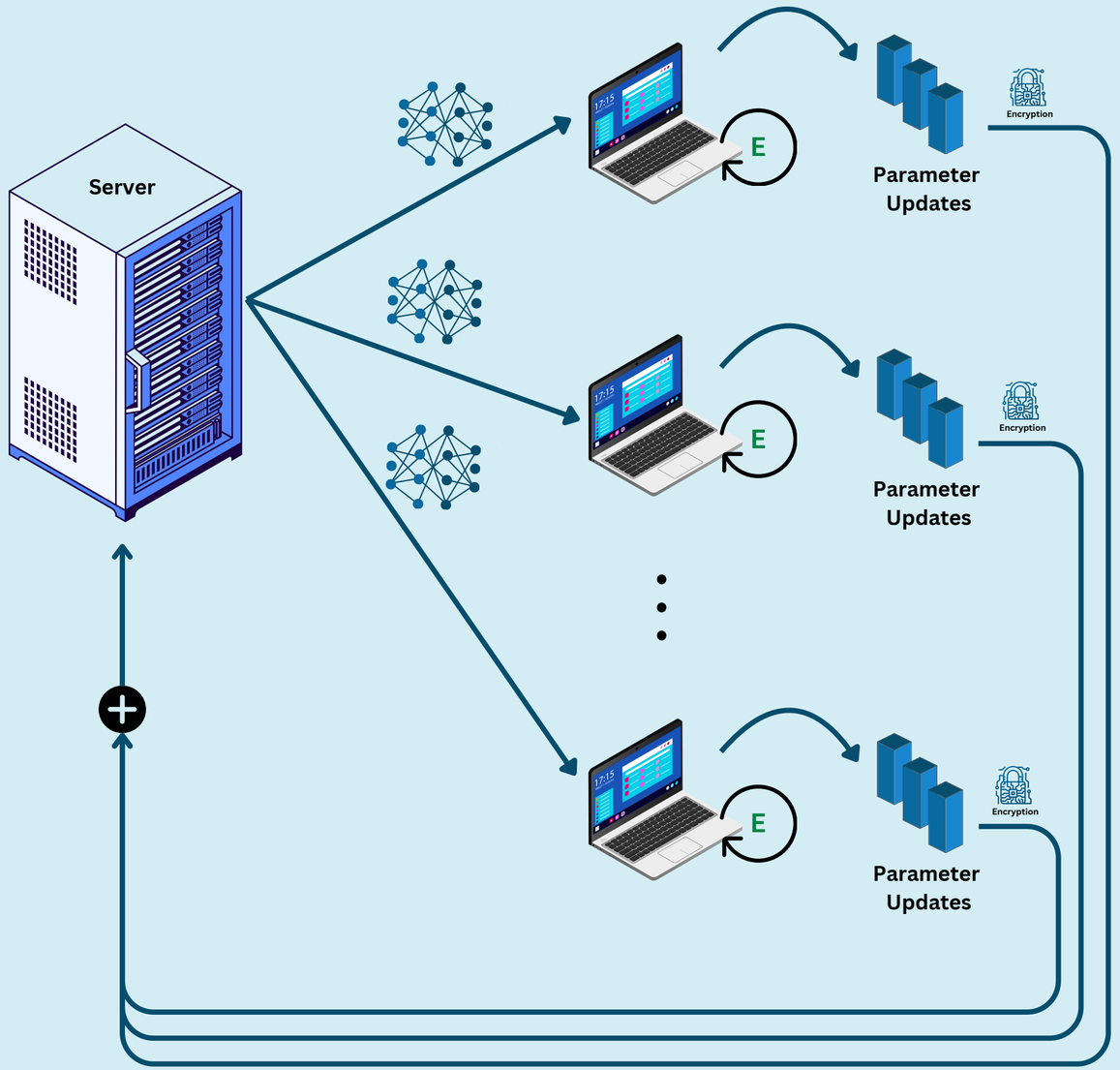
FedAvg takes approximately 3x the time for FedSGD (Internal epochs = 3 for FedAvg here)

Note: Diabetes MLP is a very small model compared to MNIST MLP/Fashion MNIST CNN so the internal training time for Diabetes MLP is less than the communication overhead, whereas in the other two the communication overhead is negligible as compared to the internal train time. Thus the anomaly in the 3x time for Diabetes MLP.

What if client data is non-uniformly distributed?

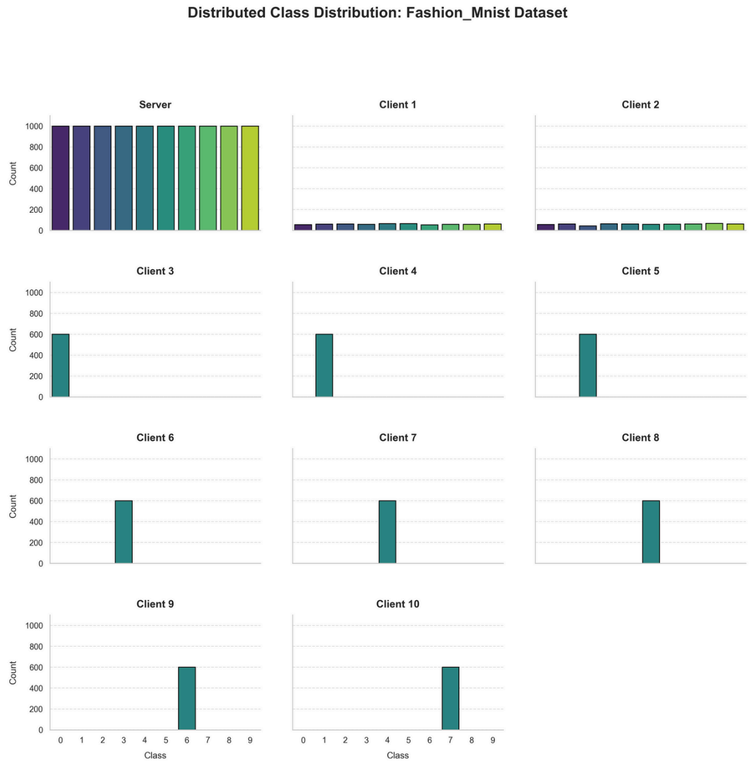
Problems to tackle

- Each client is treated *equally*
- What about weights from clients which have bad data?



FedAdp: Adaptive Model

Visualising data distribution



2 IID + 8 Non-IID (FashionMNIST)

Here number of data samples with each node = 600.

For IID nodes we randomly select 600 data samples for the original Dataset.

For Non-IID nodes we randomly select 600 data samples from a particular class in the original Dataset.

8 IID + 2 Non-IID (FashionMNIST)

Note: Since the original distribution is uniform across classes, the random selection for IID nodes also led to uniform distribution (class-wise) in IID nodes data.

Algorithm

Algorithm 3 Federated Adaptive Weighting (FedAdp)

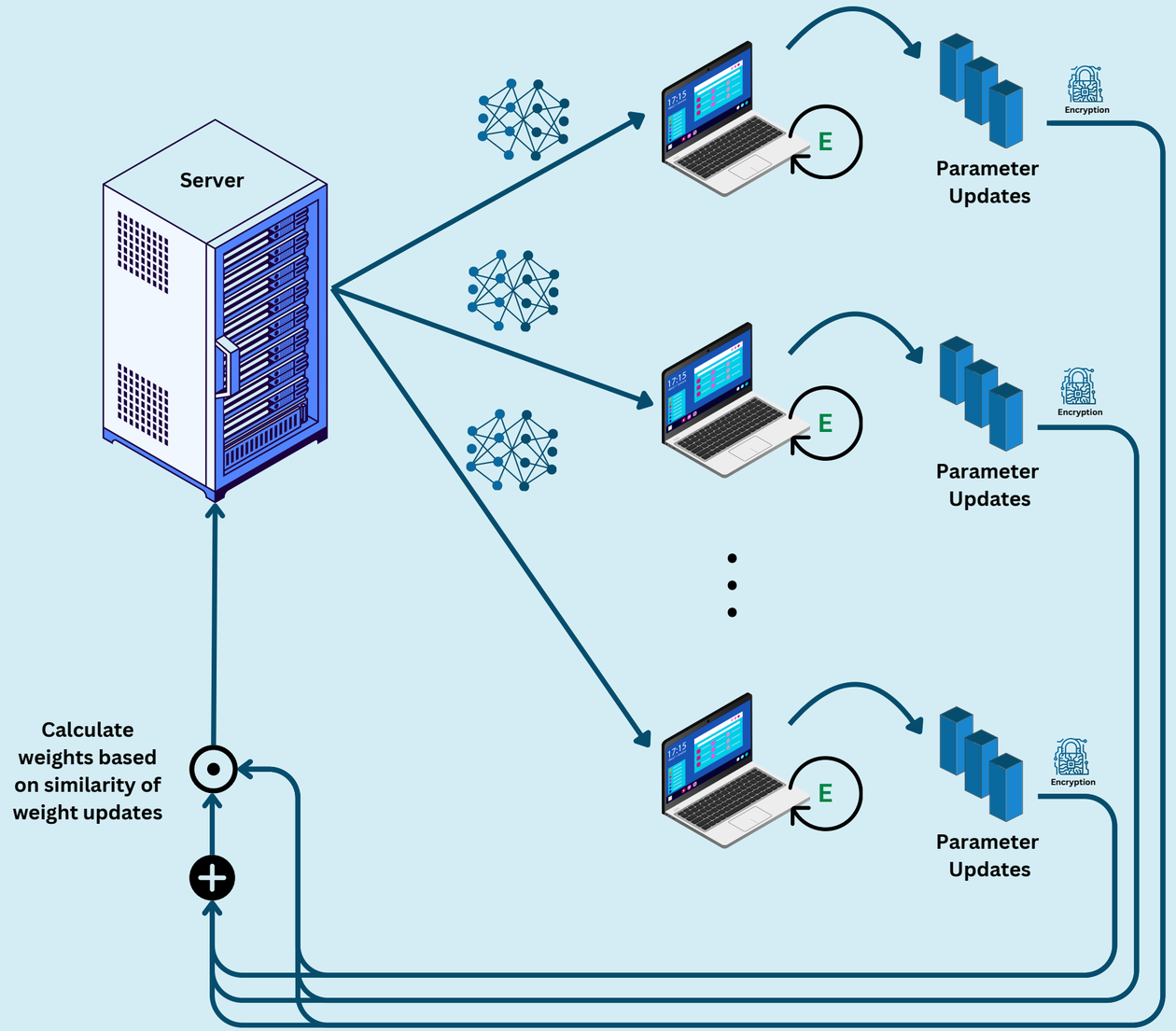
procedure FEDERATED OPTIMIZATION
Input: node set S, E, B, T, η
  Server initializes global model $\mathbf{w}(0)$, global update $\Delta(0)$,
  smoothed angle $\tilde{\theta}_i(0)$, $i \in S$
  for $t = 1, \ldots, T-1$ **do**
    for node $i \in S_t$ **in parallel do**
      $\Delta_i(t) \leftarrow$ LOCAL UPDATE$(i, \mathbf{w}(t-1))$
    end for
    $\mathbf{w}(t) \leftarrow$ GLOBAL UPDATE$(\Delta_1(t), \Delta_2(t), \cdots, \Delta_{|S_t|}(t))$
  end for
end procedure

procedure LOCAL UPDATE
Input: node index i, model $\mathbf{w}_i(t-1)$
  Calculate local updates for $\tau = D_i \frac{E}{B}$ times of SGD with step-size η on $F_i(\mathbf{w})$ and obtain $\mathbf{w}_i(t)$ using (6)
  Calculate the model difference $\Delta_i(t) = \mathbf{w}_i(t) - \mathbf{w}_i(t-1)$
  return $\Delta_i(t)$
end procedure

procedure GLOBAL UPDATE
Input: local update $\Delta_1(t), \Delta_2(t), \cdots, \Delta_{|S_t|}(t)$
  Calculate the global gradient:

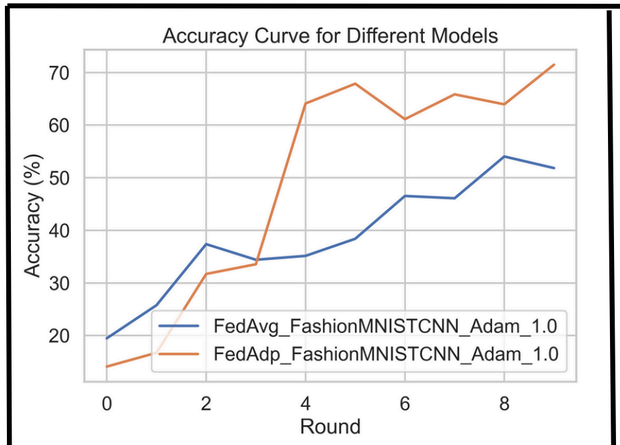
$$\nabla F(\mathbf{w}(t)) = \sum_{i=1}^{|S_t|} \left(D_i / \sum_{i'=1}^{|S_t|} D_{i'} \cdot \nabla F_i(\mathbf{w}(t)) \right), \text{ where } \nabla F_i(\mathbf{w}(t)) = -\Delta_i(t)/\eta$$

  Calculate instantaneous angle $\theta_i(t)$ by (15)
  Update smoothed angle $\tilde{\theta}_i(t)$ by (16)
  Calculate weight $\tilde{\psi}_i(t)$ for model aggregation by (20), (21)
  Update global model: $\mathbf{w}(t) = \mathbb{E}_{i,t}\left[\tilde{\psi}_i(t)\mathbf{w}_i(t-1)\right]$
  return $\mathbf{w}(t)$
end procedure

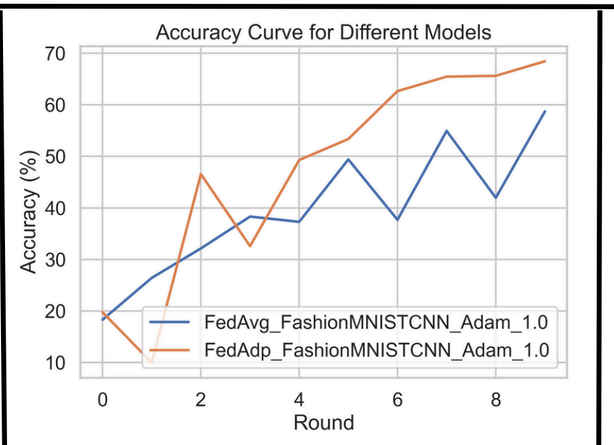


FedAvg vs FedAdp

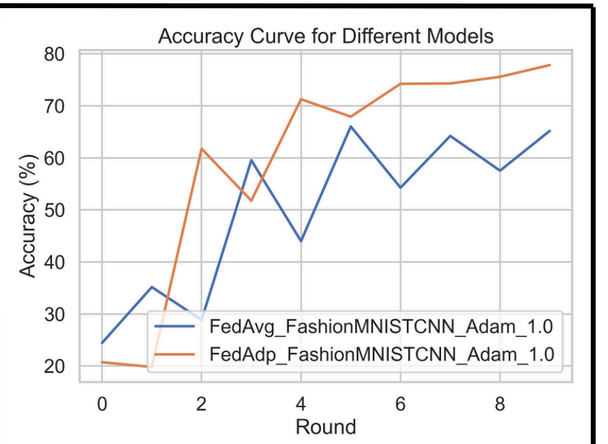
2 IID + 8 Non-IID

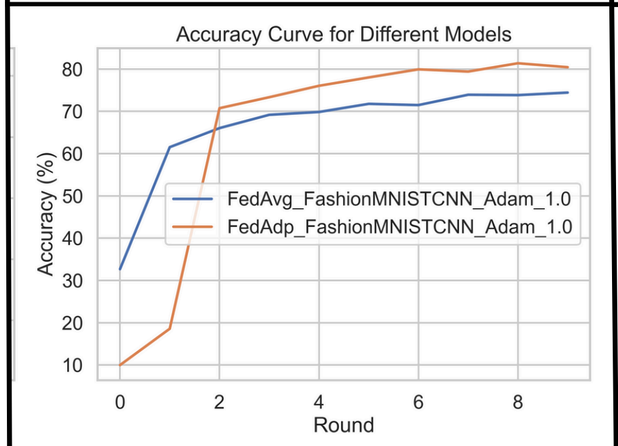


3 IID + 7 Non-IID

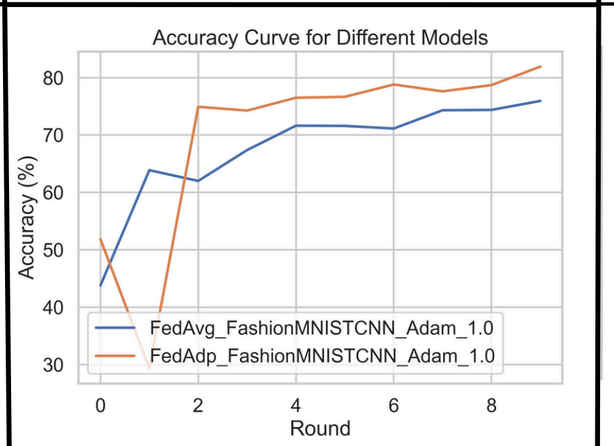


4 IID + 6 Non-IID

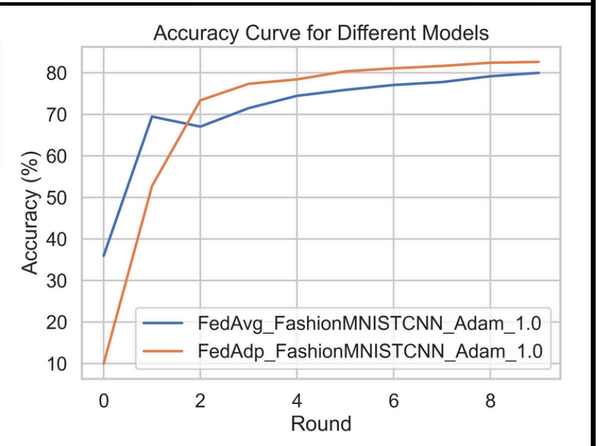




6 IID + 4 Non-IID



7 IID + 3 Non-IID



8 IID + 2 Non-IID

IID: Equal representation of each class in the data with the node.

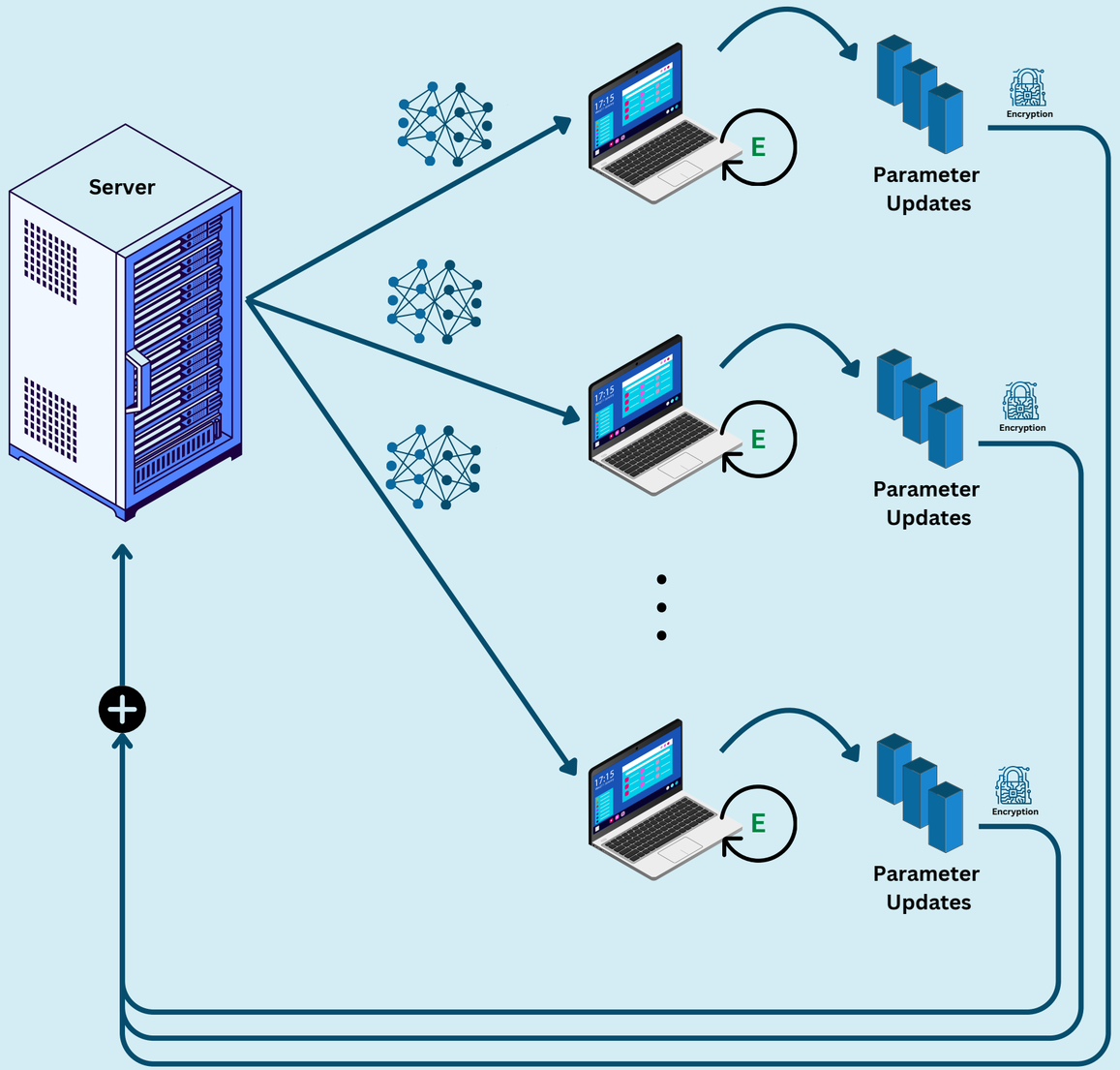
Non-IID: All samples of a single class in the data with the node.

In general FedAdp performs better than FedAvg but the disparity increases with the increase in proportion of Non-IID nodes in the training network.

What happens when clients are slow ?

Problems to tackle

- The FL server waits for *all* clients to finish local training
- Slower clients become bottleneck for faster training

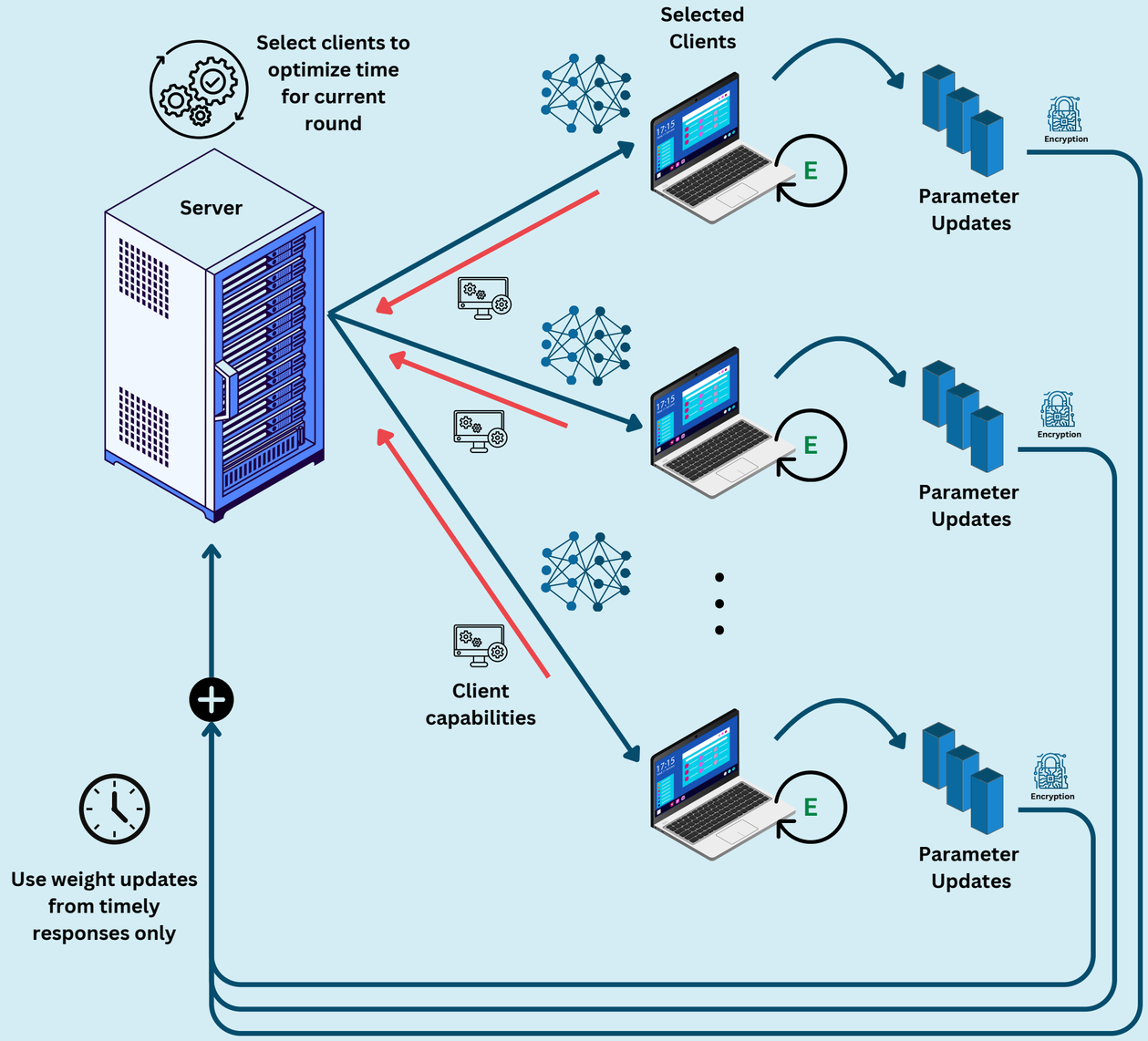


FedCS: Selective Model

Algorithm

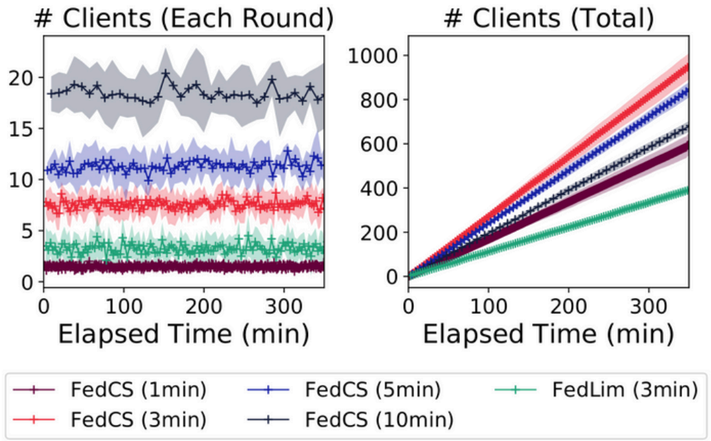
Algorithm 4 Client Selection (FedCS)

procedure FEDERATED CS
Require: Index set of randomly selected clients $\mathbb{K}'$
  Initialization: $\mathbb{S} \leftarrow \{\}, T^{\mathrm{d}}_{\mathrm{S}=\emptyset} \leftarrow 0, \Theta \leftarrow 0.$
  while $|\mathbb{K}'| > 0$ **do**
    $x \leftarrow \arg\max_{k \in \mathbb{K}'} \frac{1}{T^{\mathrm{d}}_{\mathrm{S}\cup k} - T^{\mathrm{d}}_{\mathrm{S}} + t^{\mathrm{UL}}_{k} + \max\left\{0, t^{\mathrm{UD}}_{k} - \Theta\right\}}$
    remove x from $\mathbb{K}'$
    $\Theta' \leftarrow \Theta + t^{\mathrm{UL}}_{x} + \max\left\{0, t^{\mathrm{UD}}_{x} - \Theta\right\}$
    $t \leftarrow T_{\mathrm{cs}} + T^{\mathrm{d}}_{\mathrm{S}\cup x} + \Theta' + T_{\mathrm{agg}}$
    if $t < T_{\mathrm{round}}$ **then**
      $\Theta \leftarrow \Theta'$
      add x to $\mathbb{S}$
    end if
  end while
  return $\mathbb{S}$
end procedure

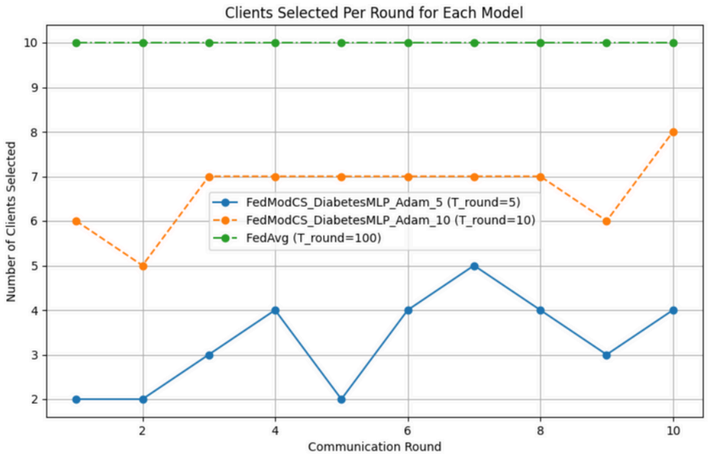


FedCS Ablation Study

Distribution of Client Capabilities



Reference plot (From paper)

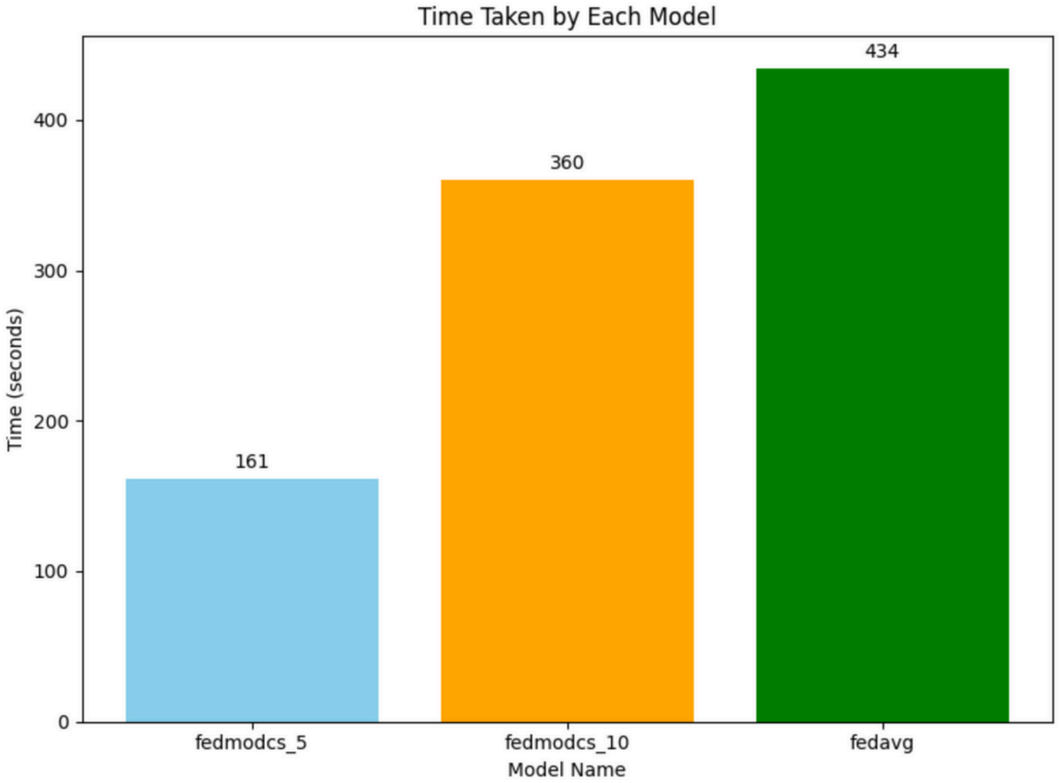


Our plot

Graph showing how number of clients selected increases with increase in T_round (tradeoff between clients involved in single round and overall training time)

FedCS vs FedAvg

- **FedCS dynamically selects clients** based on upload and update times, ensuring that each communication **round completes within the specified t_round**.
- This **avoids delays caused by slow clients**, leading to faster overall training.
- FedCS with t_round = 5 achieves accuracy same as FedAvg but in less than half the time.
- The ability to constrain round duration makes FedCS **more suitable for real-world** federated settings where **devices have heterogeneous capabilities** and time constraints.
- By adjusting t_round, FedCS allows fine-grained **control over the trade-off between speed and accuracy** depending on application needs.



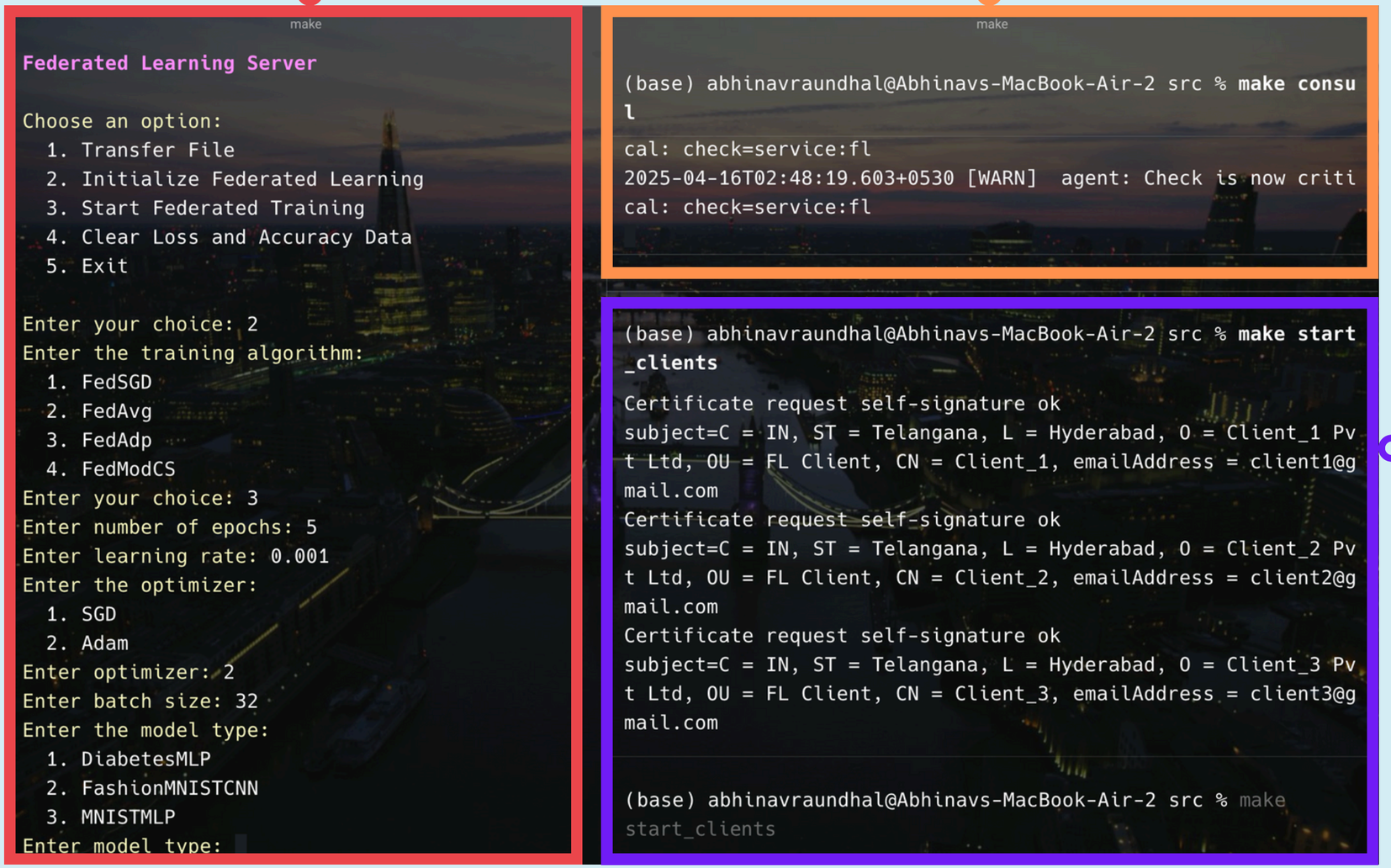
Time taken during training to reach a 95% accuracy

Demostration

FEDERATED LEARNING SERVER

CONSUL

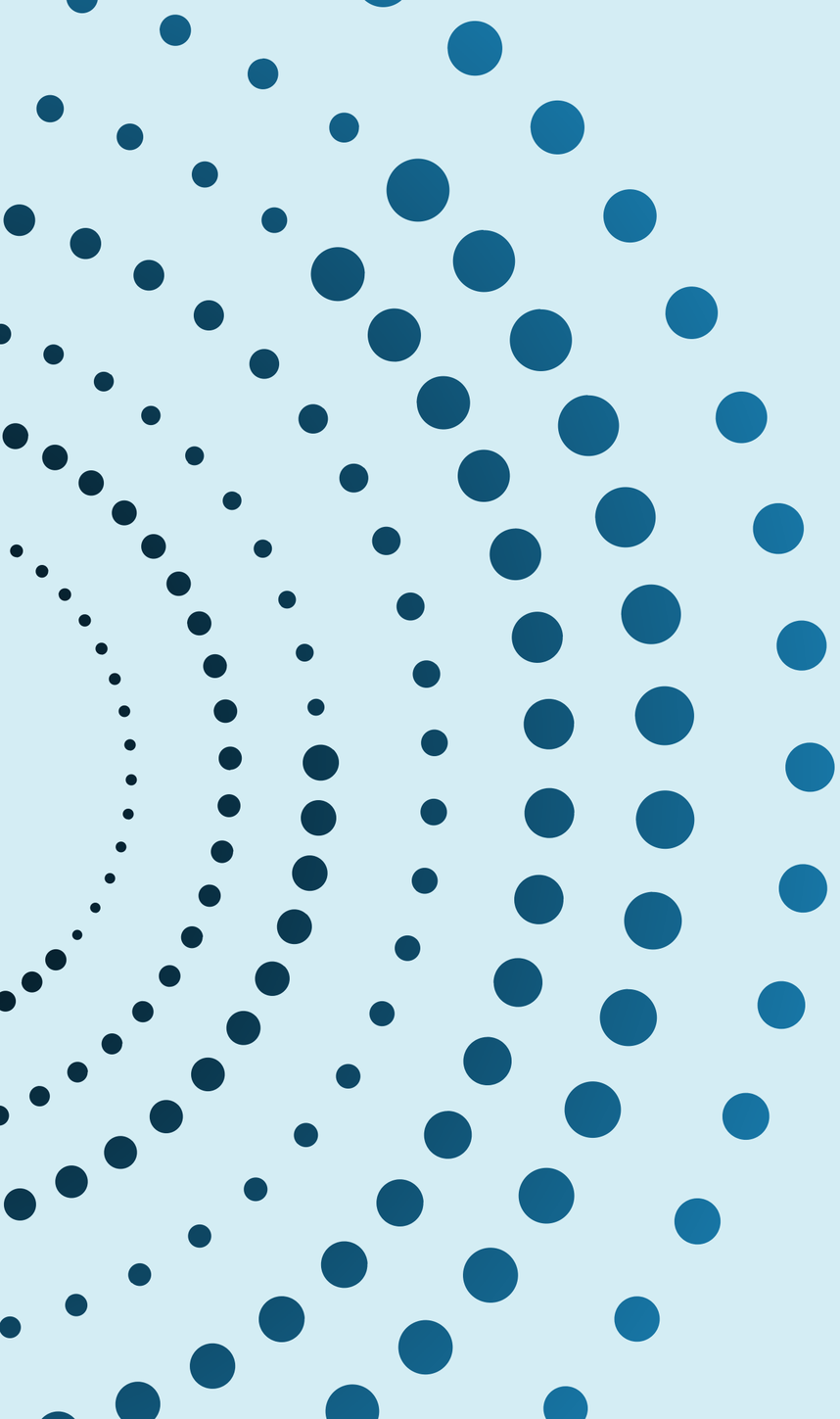
CLIENTS (Run in the background)



Contributions (All Equal)

Tasks ⬇	Abhinav	Archisha	Vinit
ML Models	R	R	C
gRPC	C	R	R
FedSGD	C	R	C
FedAvg	R	C	C
FedAdp	R	C	R
FedCS	C	R	C
Encryption	C	C	R
Ablation	R	R	R
PPT	R	R	R

R = Responsible C = Consulted



Thank you!

Github – DistLearn

Website – DistLearn