Communication-Efficient Learning of Deep Networks from Decentralized Data

MADHUPRANAVI S - ED22B032
PUSHPA CHAUDHARY - EE21B109

EE5180
INTRODUCTION TO MACHINE LEARNING

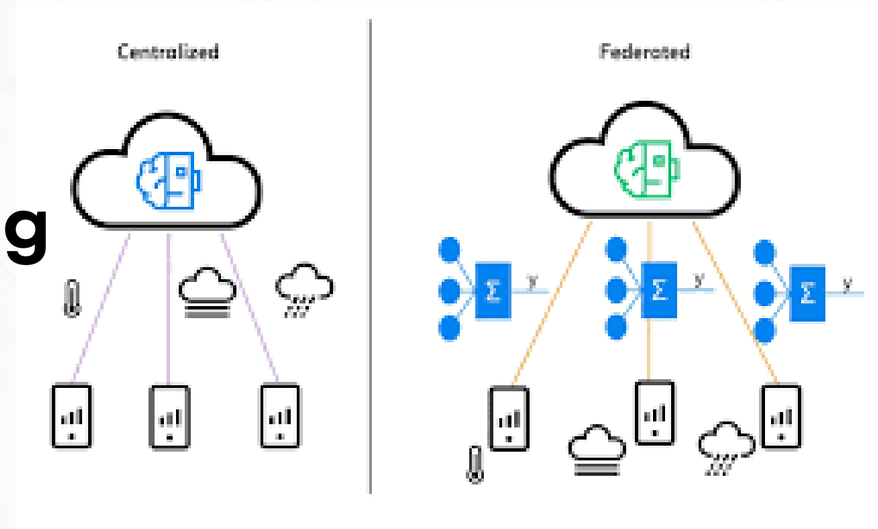
May 17, 2025

Motivation

- **Rich data** (text, images, speech) on personal devices
- Potential to improve user experience through intelligent applications
  - Image Classification: Photo popularity prediction.
  - Language Modeling: Next-word prediction on keyboards.
  - Speech Recognition: Personalized voice assistants

Context

- Centralized training poses privacy risks and scalability issues
- Challenge: Train ML models without transferring data off devices or compromising privacy
- Term coined: **Federated Learning**
- Opportunity: Leverage device computation power



Challenges

- **Non-IID data:** Data on clients is based on personal use patterns, so it differs greatly across users.
- **Unbalanced data**: Some users have much more data than others.
- **Massively distributed**: Millions of devices may participate.
- **Limited communication**: Upload speed is low, and devices are often offline.

Hypothesis

1

Iterative model averaging effectively train deep models on decentralized, privacy-sensitive data

2

Communication costs can be drastically reduced by increasing local computation

3

Model accuracy will be robust even with unbalanced and non-IID client data.

Experimental Setup

Datasets:

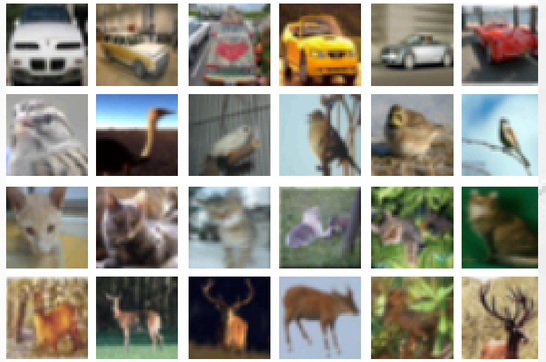
- MNIST: 100 clients, IID & non-IID
- Shakespeare: 1146 clients (roles)
- CIFAR-10: 100 clients, balanced IID
- Social Network: 500,000 clients

Models:

- 2-layer MLP, CNN, LSTMs

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.
```



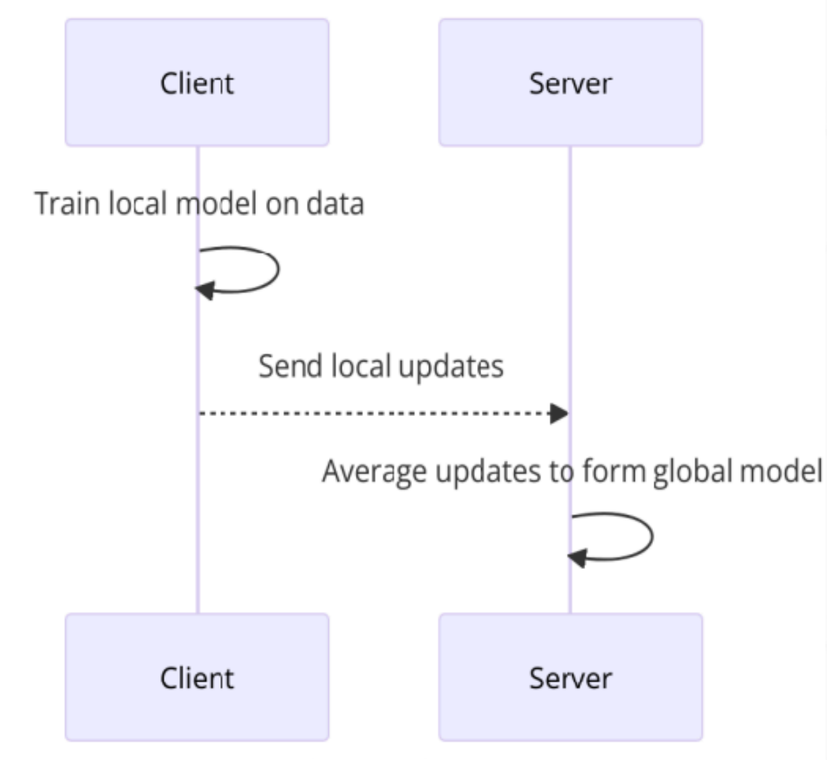
FedAvg Algorithm

Server: Initialize model → Select clients → Aggregate updates → New global model

Clients: Receive model → Local SGD → Send updates

Key parameters:

- C: fraction of clients
- E: number of local epochs
- B: local batch size



FederatedSGD

Let K be the total number of clients. Each client k has local dataset of size nk .

Total data size:

$$n = \sum_{k=1}^{K} n_k$$

Global objective function:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

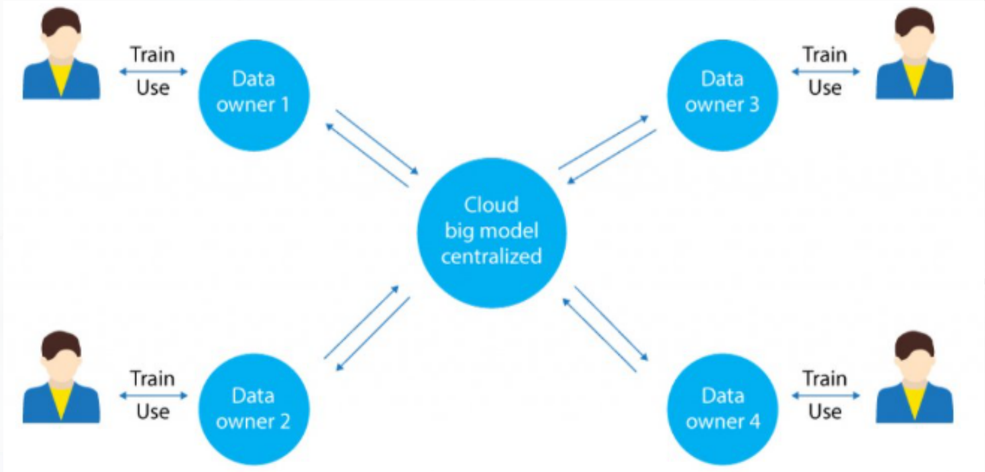
where $F_k(w)$ is the local loss at client k.

Communication vs Computation in Federated Optimization

Our goal is to use additional computation in order to decrease the number of rounds of communication needed to train a model.

Two strategies:

- Increased parallelism
- Increased computation on each client



Federated Averaging (FedAvg)

Select a fraction C of clients in each round. Each client k computes the local gradient: $g_k = \nabla F_k(w_t)$

Server aggregates weighted gradients: $w_{t+1} = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$

Each selected client performs multiple local updates, for multiple local epochs E . $w_k^{t+1} = w_k^t - \eta \nabla F_k(w_k^t)$

Server aggregates updated models: $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^{t+1}$

Pseudo Code

Server:

1. Initialize w_0
2. For each round $t = 1, 2, \ldots$
   - Select random set S_t of $m = \max(C \cdot K, 1)$ clients
   - For each client $k \in S_t$ (in parallel):
     $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
   - Aggregate: $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w)**:**

1. Split $\mathcal{P}_k$ into batches of size B
2. For $i = 1$ to E
   - For each batch b: $w \leftarrow w - \eta \nabla \ell(w; b)$
3. Return updated w

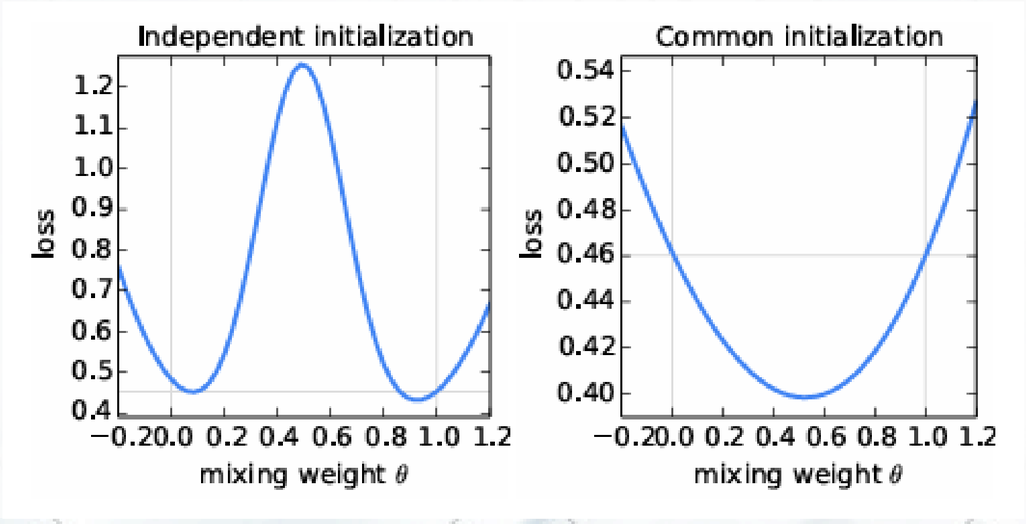
Averaging Models

Averaging models from different initializations:

$$w = \theta w_1 + (1 - \theta) w_2, \quad \theta \in [0, 1]$$

With independent initializations, this can result in high loss.

With common initialization, averaging tends to reduce loss.



Empirical Observation

If w1 and w2 are trained independently on IID datasets with common initialization:

$$w_{avg} = \frac{1}{2}w_1 + \frac{1}{2}w_2$$

Often yields better performance than either model alone.

Convergence Rate: Depends on the number of clients and data distribution.

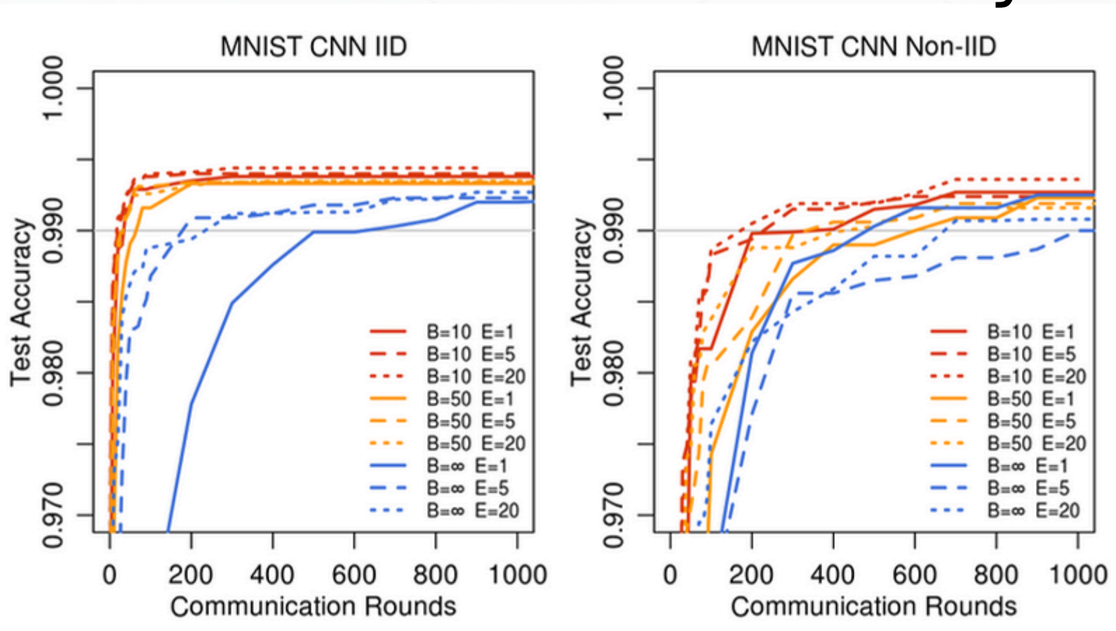
Affected by client fraction C and local update count E .

Higher E may lead to faster convergence but also higher risk of overfitting.

Results

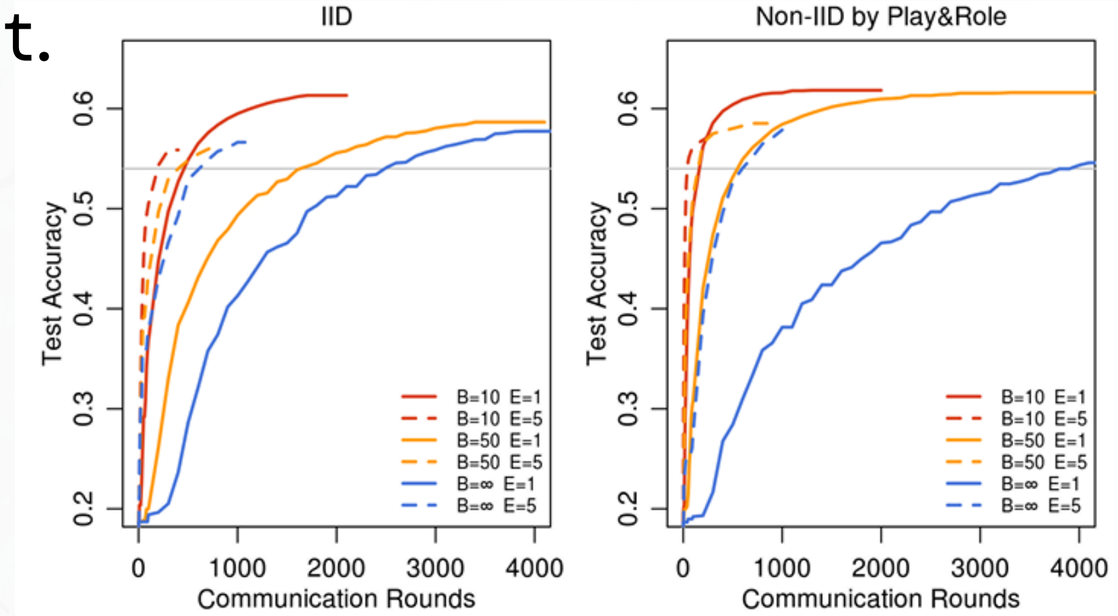
MNIST

- **Dataset**: Handwritten digits (0–9), 100 clients
- **IID Data**: FedAvg reduced communication rounds by **35×** vs. FedSGD.
- **Non-IID Data:** Still achieved **2.8–3.7×** speedup despite pathological data splits.
- FedAvg remains stable.



Shakespeare (LSTM)

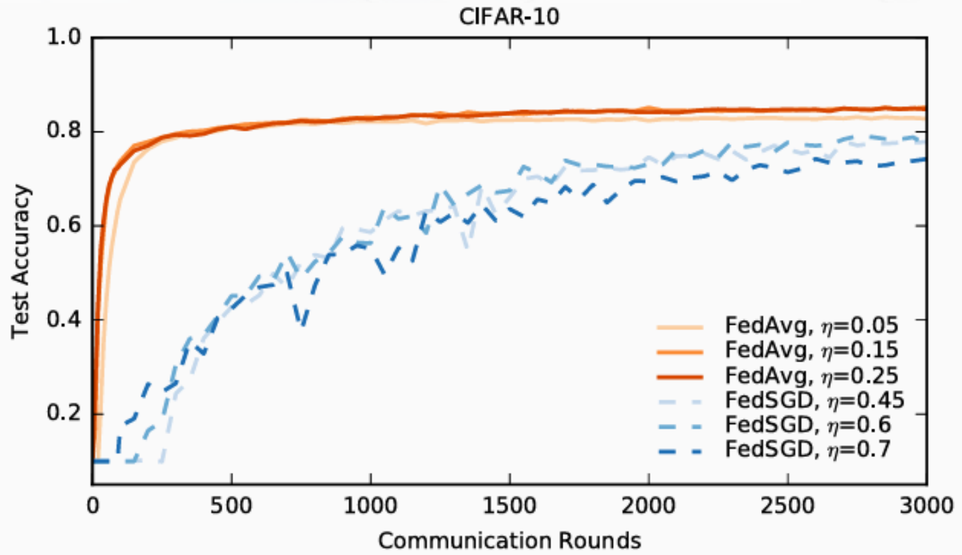
- Dataset: **1146** speaking roles from plays
- **Highly non-IID** and unbalanced
- **IID Data: 13×** improvement.
- **Non-IID Data**: reduced rounds from 3906 (FedSGD) to just 41
- Speedup up to **95×**



Results

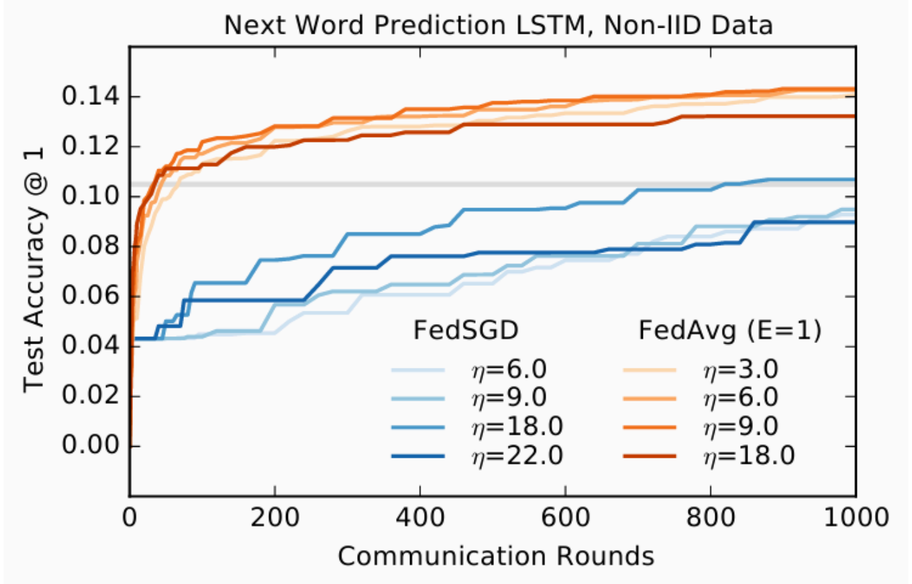
CIFAR – 10

- **Dataset:** 10-class image classification, 100 clients
- FedAvg: **85%** accuracy in **2,000** communication rounds vs **197,500** for SGD
- Speedup: **64× to 49×** depending on accuracy target
- Remarkable convergence



Social Network (LSTM)

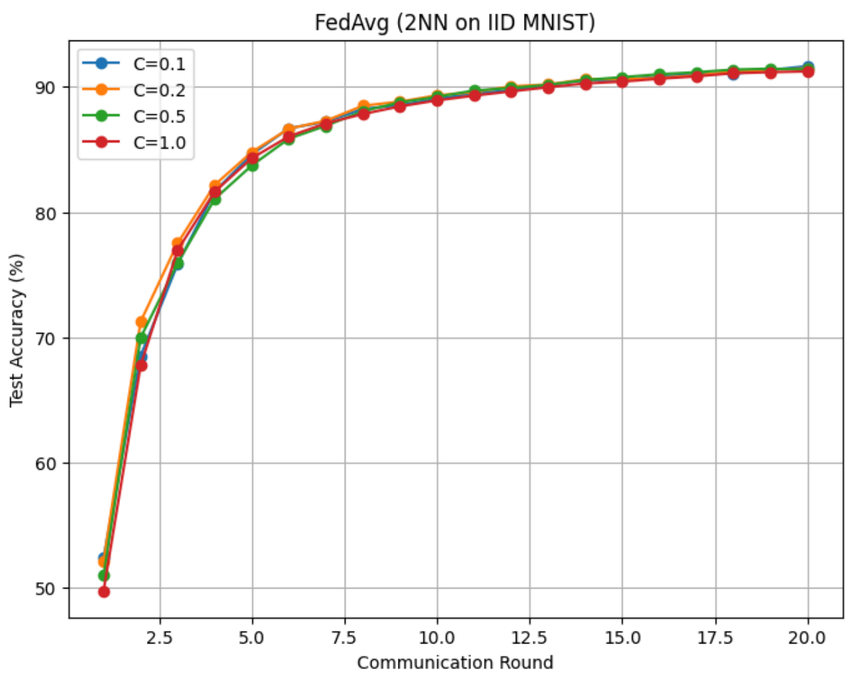
- Dataset: **500,000** clients, 10M public social media posts
- Task: **next-word prediction**
- FedAvg vs. FedSGD:
  - FedSGD: 820 rounds to reach **10.5%** accuracy.
  - FedAvg: **23×** fewer rounds (35 rounds) for same accuracy.



Results

MNSIT- Self Implementation

```
100%|██████████| 9.91M/9.91M [00:00<00:00, 56.8MB/s]
100%|██████████| 28.9k/28.9k [00:00<00:00, 1.76MB/s]
100%|██████████| 1.65M/1.65M [00:00<00:00, 14.8MB/s]
100%|██████████| 4.54k/4.54k [00:00<00:00, 4.96MB/s]Train dataset size: 60000
Test dataset size: 10000
```




```python
def client_update(client_model, optimizer, train_loader, epochs=EPOCHS):
    """
    Perform local training on client dataset.
    """
    client_model.train()
    criterion = nn.CrossEntropyLoss()
    for epoch in range(epochs):
        for data, target in train_loader:
            data, target = data.to(device), target.to(device)
            optimizer.zero_grad()
            output = client_model(data)
            loss = criterion(output, target)
            loss.backward()
            optimizer.step()
    return client_model.state_dict()  # Return updated weights
```

```python
# Constants
NUM_CLIENTS = 100  # Number of clients for simulation
EPOCHS = 1         # Number of local epochs per client per
BATCH_SIZE = 10    # Local batch size (can be varied later
LEARNING_RATE = 0.01  # Local learning rate

print(f"Simulating {NUM_CLIENTS} clients with batch size
```

```
Simulating 100 clients with batch size 10 and local epochs 1.
```

Implications of FedAvg

- FedAvg demonstrates effectiveness even in highly non-IID data environments → its robustness in realistic decentralized scenarios.
- Model averaging across diverse local datasets introduces a regularization effect, akin to dropout, which may enhance generalization.
- The algorithm encourages training directly on decentralized, privacy-sensitive devices such as smartphones, enabling practical deployment at the edge.

Key Takeaways

- FedAvg outperforms FedSGD in both convergence speed and final accuracy.
- Non-IID datasets do not always hinder performance; in some tasks (e.g., Shakespeare), they even accelerate training.
- Tuning learning rate, E, B, and C is essential for optimal performance.
- FedAvg is robust: it converges even when individual clients have vastly different local data distributions.

Challenges Faced

- Pathological non-IID partitions can still cause slower convergence (e.g., MNIST 2-shard per client case).
- Hyperparameter tuning is sensitive and non-trivial in federated settings.
- Client variability (in compute or data) can introduce instability or inefficiency.
- Communication bottlenecks persist for extremely large models or frequent updates.

Future Work

- Adaptive hyperparameter tuning for learning rates, batch sizes, and local epochs on-device.
- Study heterogeneous client systems—with differing compute, memory, and bandwidth constraints.
- Explore privacy guarantees (e.g., differential privacy, secure aggregation) in conjunction with FedAvg.
- Extend to more complex tasks: multimodal learning, federated reinforcement learning, etc.

Conclusion

- FedAvg enables scalable and communication-efficient federated learning.
- It maintains high accuracy even with minimal communication and non-IID client data.
- Future work must tackle system heterogeneity and privacy constraints for real-world deployment.

“Robustness and efficiency of FedAvg make it a strong baseline for federated optimization.”

Contribution

Sections Contributed by Pushpa Chaudhary - EE21B109:

- Introduction - Motivation, Context, Challenges, Hypothesis
- Methods - Experimental Setup, FedAvg Algorithm
- Results - MNIST, Shaksepere, CIFAR, LSTM

Sections Contributed by Madhupranavi S - ED22B032:

- Mathematical Foundation - FederatedSGD, Communication efficiency, FedAvg Algo- pseudocode, Avg -modelling, Empirical Observation
- Implications & Conclusions- Implications of FedAvg & Key Takeaways, Challenges Faced, Future Work, Conclusion

Question and Answer...

Thank You