UNIVERSITY OF HELSINKI

Seminar on Edge Intelligence

Presentation on Edge Training (Federated Learning)

By

Sabina Zaman

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” International Conference on Artificial Intelligence and Statistics, pp. 1273–1282, Apr. 2017, [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf

[2] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," arXiv preprint arXiv:1804.08333, Oct. 2018.

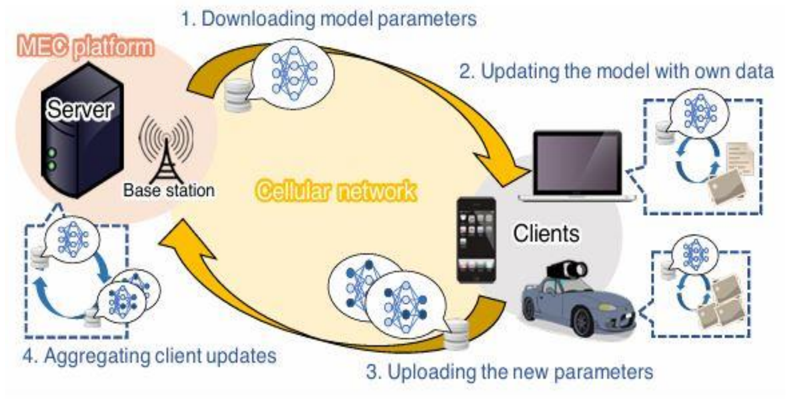
Introduction

Server: Coordinates the learning task.

Clients/ Data Source: A loose federation of participating devices. For example, IoT devices, smartphones, and autonomous vehicles with high-resolution sensors, all of which are connected to a high-speed network.

Federated learning* enables the training of machine learning models on private client data through the iterative communication of model parameters between a server and clients.

- Clients download a model from a server.
- They update the model using local data.
- Clients upload updated parameters to the server.
- The server aggregates updates to improve the model.[2]


1. Downloading model parameters
MEC platform
Server
Base station
Cellular network
2. Updating the model with own data
Clients
4. Aggregating client updates
3. Uploading the new parameters

Comparison between Data Center Training and FL

Data center training on persisted data has two issues

1. Data is privacy sensitive.
2. For supervised tasks, labels on the data can be modified because of user interaction. [1]

FL gives benefits here

1. <u>Minimal Information Sharing</u>: Only necessary updates are transmitted, not the raw data
2. <u>Ephemeral Updates:</u> The updates are temporary and contain less information than the full dataset
3. <u>Anonymized Transmission:</u> The updates do not require identifying metadata and can be transmitted through secure channels.
4. <u>Advanced Privacy Techniques:</u> FL can be combined with secure multiparty computation and differential privacy for enhanced protection. [1]

Federated Learning Optimization Issues

The optimization problem implicit in federated learning is federated optimization

- Non-IID(Independent and Identically Distributed): A client's training data is user-specific and not representative of the overall population.

- Unbalanced: Users vary in usage, leading to different amounts of local training data.

- Massively distributed: The number of participating clients is much larger than each client’s data.

- Limited communication: Mobile devices often face connectivity issues, being offline or on slow networks.

Contribution of Communication-Efficient Learning of Deep Networks from Decentralized Data [1]

<u>Ignored Practical challenges</u>

1. Dynamic client datasets that change as data is added or deleted.
2. Client availability can vary based on local data distribution for example different usage patterns between regions.
3. Unresponsive or faulty clients may never send updates or provide corrupted data.

Focus on addressing non-IID, unbalanced data, and client availability in federated optimization

Overall Optimization Objective

The goal is to minimize the loss function.

For a machine learning problem, we typically takef_i(w) = ($x_i y_i$;w), that is, the loss of the prediction on example ($x_i y_i$) made with model parameters w.

$$\min_{w \in \mathbb{R}^d} f(w) \qquad \text{where} \qquad f(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w).$$

We assume there are K clients over which the data is partitioned, with P_k the set of indexes of data points on client k, with n_k = P_k .

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w).$$

Experimental Environment [1]

1. A controlled environment that is suitable for experiments
2. A synchronous update scheme that proceeds in rounds of communication
3. A fixed set of K clients, each with a fixed local dataset. At the beginning of each round, a random fraction C of clients is selected, and the server sends the current global algorithm state to each of these clients (e.g., the current model parameters).
4. Each selected client then performs local computation based on the global state and its local dataset and sends an update to the server. The server then applies these updates to its global state, and the process repeats

Considerations [1]

1. Dominant Communication Costs: Federated optimization is constrained by limited upload bandwidth (around 1 MB/s or less), making communication costs a primary concern.
2. Client Participation Constraints: Clients generally participate only when charged, plugged in, and connected to unmetered Wi-Fi, limiting their availability and frequency of participation.
3. Limited Update Rounds: Each client is expected to engage in only a small number of update rounds per day.
4. Small On-Device Datasets: Individual datasets on clients are relatively small compared to the total dataset, which means the computation on clients is less costly compared to the communication required.
5. Powerful Client Hardware: Modern smartphones have fast processors and GPUs, making computation relatively inexpensive compared to the high communication costs.

Work Plan [1]

1. Develop Advanced Aggregation Methods: They will create improved techniques for aggregating client updates to better handle non-IID and unbalanced data.
2. Increase Computation on Clients: Implementing more complex computations on clients between communication rounds to reduce the overall number of communication rounds required.
3. Enhance Parallelism: They will utilize more clients in parallel during each update round to distribute the computational load and further decrease communication frequency.

They are going to update the basic Federated Averaging Algorithm

The Federated Averaging Algorithm Basic Version (FedSGD)

Gradient Computation at Client k

$$g_k = \frac{1}{n_k} \sum_{i \in P_k} \nabla f_i(w_t)$$

g_k -> the gradient computed by client k.
n_k -> the number of data points on client k.

P_k -> the set of indexes of data points on client k.

$\nabla f_i (w_t)$ -> the gradient of the loss function for data point i at model parameters w_t .

Server Aggregation

$$w_{t+1} = w_t - \eta \frac{1}{K} \sum_{k=1}^{K} g_k$$

w_{t+1} -> the updated global model parameters.

η -> the learning rate.

K -> the number of clients.

The Federated Averaging Algorithm Updated Version [1]

Key Parameters

1. Fraction of Clients (C): Determines how many clients are selected for each round. Global Batch Size = C * Total Data Size. If C =1, it means all clients are used, and the gradient update is performed over the entire dataset (full-batch gradient descent).

   Randomly select a fraction C of clients in each round.

2. Number of Local Updates (E): Controls how many times each client updates its model locally before sending it to the server.
3. Minibatch Size (B): Determines the size of the minibatch used for local updates on each client.

Each selected client performs E updates on their local data using minibatches of size B.

The server aggregates the updated models from selected clients to form the new global model.

The Federated Averaging Algorithm Updated Version (FedAvg) [1]

Local Updates at Client k: Each client performs E local updates on its data before sending the updated model to the server.

w_{t+e}^{k} -> the model after e local updates on client k.
B -> the minibatch size.
$\nabla f_i(w_t)$ -> the gradient of the loss function for minibatch data point i at model parameters w_t.

$$w_{t+e}^{k} = w_t - \eta \frac{1}{B} \sum_{b=1}^{B} \nabla f_i(w_t)$$

Aggregation of Local Models: After all selected clients have performed their local updates, the server aggregates these models w_{t+1}

w_{t+1} -> the new global model.
n -> is the total number of data points across all clients.
n_k -> the number of data points on client

$$w_{t+1} = \frac{1}{n} \sum_{k=1}^{K} n_k w_{t+e}^{k}$$

Experiment and Result Analysis[1]

In the experimental section focuses on evaluating the FedAvg algorithm across multiple tasks, datasets, and configurations.

A. Experimental Objectives

1. Communication Efficiency: Reduction in communication rounds but needs to find out whether it is needed to reach a target accuracy.
2. Parallelism: The effect of increasing the number of clients participating per round (client fraction, **C**).
3. Local Computation: Investigating how increasing local epochs (**E**) and varying minibatch sizes (**B**) impact convergence speed and communication efficiency.
4. FedAvg vs. FedSGD: Comparing the performance of **FedAvg** to the baseline FedSGD approach, both in terms of test accuracy and convergence rates.

Experiment and Result Analysis[1]

B. Datasets Used and Setup:

1. **MNIST Dataset**:
   - **Task**: Image classification.
   - **Models**:
     1. **2NN** (Two-layer neural network with 199,210 parameters).
     2. **CNN** (Two 5x5 convolutional layers followed by pooling, with 1,663,370 parameters).
   - **Data Partitioning**: Two types of partitioning were tested:
     1. **IID**: Data randomly shuffled and distributed equally across 100 clients.
     2. **Non-IID**: Each client received data from only two digit classes, creating highly unbalanced data distribution across clients.
2. **Shakespeare Dataset**:
   - **Task**: Language modeling (next-character prediction).
   - **Model**: Stacked character-level LSTM (866,578 parameters).
   - **Data Partitioning**: Clients represented different speaking roles from Shakespeare's works, creating highly **non-IID** and unbalanced data (some roles had many lines, others had few).

3. CIFAR-10 Dataset:
- **Task**: Image classification on 32x32 color images (10 classes).
- **Model**: CNN with 106 parameters (based on a TensorFlow tutorial).
- **Data Partitioning**: IID partitioning of 500 training examples per client across 100 clients.

4. Social Network Text Dataset:
- **Task**: Predicting the next word using a large social network text dataset.
- **Model**: LSTM model with 4,950,544 parameters (trained on a vocabulary of 10,000 words).
- **Data Partitioning**: Grouped by individual users, simulating highly non-IID data with 500,000 clients, each having at most 5,000 words.

Experiment and Result Analysis[1]

C. Key Results

1. **Impact of Client Fraction (C)**.
    - Increasing the client fraction **C** improves convergence rates, especially when using smaller minibatch sizes (**B = 10**).
    - **Smaller batches** and **more clients per round** are particularly useful in the **non-IID** case, where a higher **C** leads to better generalization and faster convergence.

2. Increasing Local Computation (E):

- Adding more **local SGD updates** per client per round leads to a **dramatic decrease** in the total number of communication rounds required to reach target accuracy.
- The CNN model achieved a **35x speedup** on IID MNIST data when increasing local computation, while the **2NN model** saw a **46x speedup**.
- On **non-IID data**, the speedup was smaller but still substantial, indicating FedAvg's ability to handle pathological data distributions well.

UNIVERSITY OF HELSINKI

Experiment and Result Analysis[1]

3. FedAvg vs. FedSGD:

- For **CIFAR-10**, FedAvg reached a test accuracy of **85%** after only **2,000 communication rounds**, while FedSGD required **6,600 rounds** to achieve a similar accuracy.
- On the **Shakespeare language model task**, FedAvg outperformed FedSGD significantly. **FedAvg** required only **35 communication rounds** to achieve 10.5% test accuracy, whereas **FedSGD** took **820 rounds**.
- The general trend showed that **FedAvg** achieves better test accuracy in fewer communication rounds, and with reduced variance, making it more communication-efficient than FedSGD.
- FedAvg performs far better in real-world scenarios like next-word prediction, where the data is naturally non-IID and unbalanced.

4. Can We Over-Optimize? (Large E):

- For some models, especially in the **later stages of convergence**, training with large values of **E** led to plateauing or divergence (e.g., Shakespeare LSTM). However, for other tasks (like the **MNIST CNN**), increasing **E** showed no significant degradation.
- This suggests that the optimal value of **E** may vary based on the model and dataset, and should be adjusted as training progresses (decaying **E** in later stages, similar to decaying learning rates).

UNIVERSITY OF HELSINKI

Experiment and Result Analysis[1]

D. Key Observations:

1. **FedAvg's Robustness**:
   - FedAvg is highly robust even when handling pathological, non-IID data with unbalanced roles. with minimal divergence.
2. **Communication Efficiency**:
   - FedAvg reduces communication rounds significantly compared to FedSGD, converging faster on large-scale datasets, making it more practical for federated learning on mobile devices and distributed systems.
3. **Trade-offs between Computation and Communication**:
   - Increasing local computation (larger **E** and smaller **B**) can reduce communication costs, but if **E** becomes too large, the method risks plateauing or divergence in certain tasks.

Conclusion[1]

Effectiveness of FedAvg:

FedAvg demonstrates that federated learning can be practical and efficient, effectively training high-quality models with relatively few communication rounds across various architectures, including multi-layer perceptrons, convolutional neural networks (CNNs), and LSTMs.

Future Directions:

Enhancing privacy through methods like differential privacy, secure multi-party computation, or their combination is a promising area for future research, particularly for synchronous algorithms like FedAvg.

Federated Learning Optimization Issues (Repeat)

The optimization problem implicit in federated learning is federated optimization

- Non-IID(Independent and Identically Distributed): A client's training data is user-specific and not representative of the overall population.

- Unbalanced: Users vary in usage, leading to different amounts of local training data.

- Massively distributed: The number of participating clients is much larger than each client’s data.

- Limited communication: Mobile devices often face connectivity issues, being offline or on slow networks.

Ref Paper[2]

This paper

- Directly addressed: unbalanced users, limited communication and
- Partially addressed: massively distributed issue.

They are going to make contributions on

- Introduction of FedCS Protocol: This protocol aims to improve FL's efficiency by actively managing the resources of heterogeneous clients in a cellular network.

The FedCS will deal with issues like

- Client Selection Problem: FedCS will set deadlines for model downloads, updates, and uploads.
- Improved Training Efficiency: FedCS reduces the time required to train ML models compared to the original FL protocol by optimizing how clients participate.

FEDERATED LEARNING WITH CLIENT SELECTION

A. Assumptions

1. Mobile Edge Computing (MEC) platform consists of a server and a base station (BS) that manages the Federated Learning (FL) process. The MEC platform leverages wireless networks during times of low congestion to handle large machine-learning models
2. Resource blocks (RBs), a unit of bandwidth resources in LTE (long term evolution), are limited. If multiple clients upload model parameters simultaneously, the bandwidth is shared, reducing throughput for each client.
3. The modulation and coding scheme (MCS) for radio communication is assumed to be well-optimized for each client based on their channel conditions. clients with poor wireless conditions will have a slower data transmission rate.
4. Despite the differences in throughput, the authors assume that the channel state and throughput are stable during the process.

FEDERATED LEARNING WITH CLIENT SELECTION

B. FedCS Protocol

1. **Initialization**: Initializing the global model.
2. **Resource Request**: The MEC operator randomly selects a subset of clients and asks them to provide information about their **resource availability**
3. **Client Selection**: Based on the information, the MEC operator selects the clients that can complete the update and upload steps **within a certain deadline**.
4. **Distribution**: The selected clients are sent the global model parameters via **multicast transmission** from the BS, which is bandwidth-efficient as it sends the same data to multiple clients.
5. **Scheduled Update and Upload**: The selected clients update the model using their local data and then **upload** the new parameters back to the server.
6. **Aggregation**: The server aggregates the updated models from clients to improve the global model.

Steps are repeated until the desired performance is reached or a final deadline arrives.

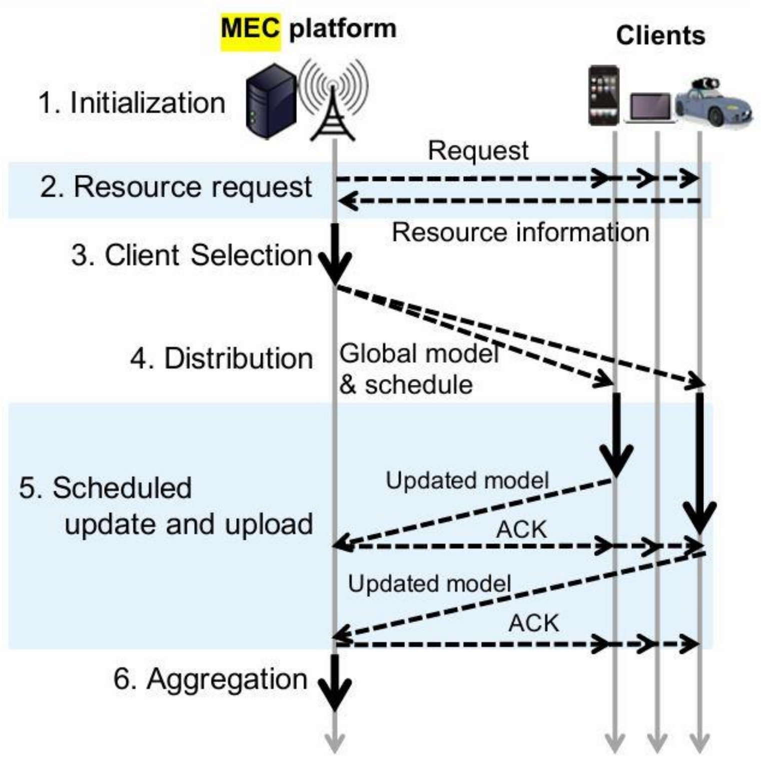


Fig. 2. **Overview of FedCS Protocol**. Solid black lines denote computation processes while dashed lines indicate wireless communications.

FEDERATED LEARNING WITH CLIENT SELECTION

C. Algorithm for Client Selection Step

1. Greedy Algorithm: Instead of trying every possible combination the algorithm iteratively selects the client that takes the least time for model updates and uploads until the deadline for the round is reached. The greedy approach helps optimize client participation by minimizing delay while maximizing the number of updates.T
2. Tuning the Deadline (Tround): The round deadline (Tround), determines how long the system waits before aggregating updates from clients. Longer deadlines allow more clients to participate but can reduce the number of updates within the overall final deadline. The paper emphasizes the importance of tuning Tround to strike the right balance between the number of clients participating in each round and the overall performance improvement of the global model.

Result Analysis

A. The Environment: A Simulated Environment MEC setup was simulated, consisting of an edge server, base station (BS), and 1000 clients.

B. Experimental Setup of ML Tasks: The experiments were based on object classification tasks using two public datasets: CIFAR-10 and Fashion-MNIST.

- **IID (Independent and Identically Distributed)**: Clients received randomly sampled data from the entire dataset.
- **Non-IID**: Clients received data from specific subsets (only 2 out of the 10 categories), which made the task more challenging.

The model used was a convolutional neural network (CNN) with six convolutional layers and three fully connected layers.

C. Global Models and Their Updates: The CNN was updated using stochastic gradient descent with a mini-batch size 50 and a learning rate of 0.025.

- Time limits were set for each round (Tround), with values ranging from 1 to 10 minutes, while the total time for all training rounds was set to 400 minutes.

Result Analysis

D. Metrics for Evaluation

- **Time of Arrival (ToA) at Specific Accuracy Levels**: Time taken to reach certain accuracy levels like 50%, 75%, and 85% on testing datasets.
- **Final Accuracy after 400 Minutes of Training:** Accuracy achieved on the test datasets by the final deadline.

E. Results in IID Settings

FedCS outperformed the baseline **FedLim** in both the CIFAR-10 and Fashion-MNIST tasks. Specifically, FedCS reached:

- 75% accuracy on CIFAR-10 76.5 minutes earlier than FedLim.
- 85% accuracy on Fashion-MNIST 33.3 minutes earlier.

FedCS involved more clients in training rounds (7.7 clients on average) compared to **FedLim** (3.3 clients), improving training efficiency.

Final Accuracy for FedCS was higher than FedLim on both datasets, indicating better model performance by incorporating more clients per round.

Result Analysis

F. Effect of Tround (Round Time Limit)

Varying the time limit (Tround) for each training round showed that neither too short nor too long deadlines worked well.

- **Short deadlines** (e.g., 1 minute) limited the number of clients, leading to lower accuracy.
- **Long deadlines** (e.g., 10 minutes) reduced the number of rounds, which also degraded performance.

A dynamic selection of Tround could optimize client involvement and improve performance.

G. Results in Non-IID Settings

- FedCS continued to outperform FedLim in the more challenging **Non-IID** scenario, achieving higher accuracy levels in both CIFAR-10 and Fashion-MNIST.
- However, both methods showed reduced accuracy and higher variance due to the non-IID data distribution, indicating the added difficulty.
- Increasing the number of clients per round or using model compression techniques could help mitigate these issues in future work.

UNIVERSITY OF HELSINKI

Conclusion

Overall Results

FedCS outperformed existing FL protocols, delivering faster training times and higher accuracy across both IID and Non-IID data distributions.

Future Work:

Explore scaling FedCS for larger models and dynamic scenarios with fluctuating resources and update times.

Conclusion:

FedCS is an efficient FL solution for MEC environments, effectively handling heterogeneous clients and reducing training time while maintaining high accuracy.

UNIVERSITY OF HELSINKI

Thank You
Any Questions?

NB. I used ChatGPT 3.5 to restructure the sentences to correct grammatical and syntactical errors.