POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Dynamic Selection Techniques for Federated Learning

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Andrea Restelli

Advisor: Prof. Luciano Baresi

Co-advisor: Tommaso Dolci

Academic year: 2022-2023

1. Introduction

Machine Learning (ML) is a specialized field in Artificial Intelligence (AI) and computer science, using data and algorithms to replicate human learning processes. Its prevalence is driven by diverse datasets, cost-effective computational processing, and affordable data storage. ML applications span computer vision, e-commerce recommendations, healthcare imaging, and self-driving cars.

Deep learning's rise in ML has led to a dependence on extensive datasets, posing challenges in data transmission and storage, including network latency and privacy concerns. Federated Learning (FL) addresses these challenges by training algorithms across decentralized clients, keeping local data within devices. Participating devices retrieve the model from a central server, conduct local model training with their respective data, and subsequently communicate the resultant model back to the server. These model weights are then aggregated, and the process can repeat in subsequent iterations, referred to as "rounds," until a satisfactory level of accuracy is attained.

While numerous tools support federated learning, no standard has emerged, and gaps exist in analyzing state-of-the-art techniques and tools.

2. Problem statement

FL offers distinct advantages over traditional ML in distributed settings. FL enhances privacy by allowing local model training on decentralized devices, reducing the need for data sharing and minimizing privacy risks. Additionally, it reduces communication overhead as data remains on the local device, alleviating network congestion.

However, FL introduces its own set of complexities. One of the distinguishing challenges of FL is the presence of statistical and system heterogeneity.

- Statistical heterogeneity arises from the non-IID (non-independent and identically distributed) nature of data on different devices, coupled with significant variations in the sizes of local data samples, influenced by individual user behaviors. This diversity in data distribution can impede model convergence compared to the homogeneous centralized dataset used in traditional ML.
- System heterogeneity encompasses variations in device attributes such as computational capabilities, memory, energy availability, and environmental factors like network speed and reliability. In contrast to the relatively uniform and robust machines

used in classic ML, FL contends with diverse and less predictable system conditions.

To address these challenges, specialized platforms and algorithms are imperative. Considering factors such as quality and size of local data, computational resources, and device availability can enhance client selection, mitigating the impact of slower clients and refining the learning process.

Existing solutions have tried to address these challenges by means either of static solutions that do not take into account the evolution of the learning over the duration of the process or either solutions that do not take into account system heterogeneity.

Lastly, experimentation is difficult, given the fact that setting up a realistic federated network for experimentation presents notable challenges and costs.

3. Contribution of the thesis

The goal of this thesis is to address the challenges outlined in Section 2, exploring methodologies and innovations to maximize the potential of federated learning while tackling its inherent complexities.

This thesis has three primary objectives: (1) analyze available FL frameworks and select the most promising one, (2) extend it with algorithms for dynamic client selection during training, and (3) further extend it with algorithms enabling resource-aware workload allocation to clients during training.

A significant contribution of this thesis is the use of an open-source framework, prioritizing reproducibility and extendability. As highlighted in Section 2, experimentation poses a critical challenge in federated learning. Thus, implementing the solution in an environment favorable to easy experiment reproduction in a simulated federated setting is crucial. At the same time, to obtain meaningful results, the experimental environment should allow reproducing conditions close to the real ones.

The work concludes with an experimental phase to assess the impact of the introduced elements compared to state-of-the-art techniques. This experimental phase aims to closely replicate a real federated environment and is designed for ease of future reproducibility and extension.

4. Framework choice

Establishing a realistic federated network for experimentation poses significant challenges and costs, especially in dealing with the inherent heterogeneity and vast scale of real-world environments. Replicating such diversity demands acquiring or renting numerous devices, incurring significant expenses. Additionally, geographic distribution introduces varying communication times, posing logistical challenges. Physical emulation mirrors real conditions but lacks flexibility to simulate diverse network states. Hardware-dependent outcomes hinder reproducibility due to changes in capabilities and network conditions.

Numerous federated learning frameworks strive to navigate and mitigate these challenges.

TensorFlow Federated (TFF) facilitates open research in federated learning, offering low-level Federated Core APIs and high-level Federated Learning APIs for model integration. *FATE* is a business-ready framework supporting secure computation protocols. While feature-rich, its extensive modules can be complex, and usability improvements are needed. *FedJAX*, a JAX-based library, prioritizes ease of use for researchers in developing and evaluating federated algorithms. However, it lacks widespread popularity and active maintenance. *FedML* provides a comprehensive platform with versatile federated learning simulation, but its complexity may pose a challenge for users. *PySyft* extends beyond FL, supporting remote data science with privacy measures. Despite its potential, documentation gaps hinder clarity. Each framework has its strengths, but challenges in usability and documentation persist.

Among these frameworks, we selected *Flower* [1] for the following reasons:

- **User-Friendly:** Flower prioritizes ease of use with well-maintained documentation, tutorials, and baselines, aligning perfectly with the goal of enabling easy reproduction of results by other researchers.
- **Extendability:** Flower offers flexibility through numerous easily extendable classes, allowing a focus on core algorithms without navigating complex structures. This sets it apart from frameworks like TensorFlow Federated with intricate and poorly documented core APIs.

- **Versatile Capabilities:** Flower stands out as one of the most versatile frameworks, accommodating various client types and supporting major machine learning frameworks, making it suitable for diverse tasks, from mobile to edge devices.
- **Built-in Simulation Engine:** Flower's simulation engine facilitates quick prototyping and verification of strategies, allowing experimentation with different configurations without limitations.
- **Active Community and Popularity:** Flower's popularity and active community support ensure ongoing updates and maintenance, vital for long-term reproducibility and research advancement.

In summary, Flower excels in user-friendliness, extendability, versatility, simulation capabilities, and community engagement, making it the ideal platform for advancing federated learning research in this thesis.

5. Implementation

As highlighted in Section 3, this thesis focused its contributions in two directions: strategies for dynamic selection of clients and strategies for resource-aware workload allocation in federated learning.

5.1. Dynamic selection

Five state-of-the-art strategies for dynamic client selection were implemented and integrated into the Flower framework to facilitate comparison and to ensure straightforward reproducibility.

FedAvg

FedAvg [4] strategy serves as a benchmark for client selection in federated learning. Consider a setting with a predetermined set of K clients. Each client k has a unique local dataset of size n_k. At the beginning of every round, this strategy consists of selecting a random subset S_t of clients, which represents a fraction C of the total clients. *Flower* provides a built-in *FedAvg* implementation among its baselines, ensuring seamless integration for new projects that wish to benchmark against *FedAvg*. We leveraged this built-in strategy, adapting it to our specific clients and data-loading pipeline. This serves as our baseline for comparing new techniques.

Dynamic sampling

This strategy overcomes *FedAvg*'s static nature by dynamically adjusting the number of clients selected at each round [2]. It starts with a high sampling rate, gradually reducing it with each round to accelerate initial convergence. As the federated model matures, fewer clients participate, conserving communication resources. The exponential decay rate (β) governs the dynamic sampling, ensuring fewer parameter transmissions than static methods over rounds. The dynamic subsampling, represented as $R(t, \beta) = \frac{1}{\exp(\beta t)}$, yields a dynamic rate $c = \frac{C}{\exp(\beta t)}$, differentiating it from static techniques. We extended Flower to include Dynamic Sampling by creating a custom *Strategy*.

pow-d

The base variant of the *Power of Choice* techniques aims to select clients dynamically based on their local losses, prioritizing those with higher loss for faster convergence [3]. The strategy involves three phases for server-side client selection:

1. **Sample Candidate Client Set:** The central server samples a set A of d clients, choosing each client k with probability p_k, the fraction of data at the k-th client.
2. **Estimate Local Losses:** Server sends the global model $w^{(t)}$ to clients in A for computation of local loss $F_k(w^{(t)})$.
3. **Select Highest Loss Clients:** The server forms the active client set $S^{(t)}$ by choosing the top $m = \max(CK, 1)$ clients with the highest $F_k(w^{(t)})$, with ties broken at random.

The implementation of this technique in Flower involved creating a custom Server, Strategy, and other classes for different variants (*pow-d*, *cpow-d*, *rpow-d*) of the *Power of Choice* family.

cpow-d

The *Power of Choice* base strategy (*pow-d*) has two primary drawbacks:

- Requires a preliminary phase where each client evaluates the entire local dataset, leading to increased computational overhead.

- Involves all clients communicating their local losses to the server in every round, introducing additional communication cycles and increased costs.

To address these issues, a more computationally efficient variant, referred to as *cpow-d*, is proposed. Unlike *pow-d*, *cpow-d* has clients compute F_k on a mini-batch of b samples randomly selected from B_k. This enhances efficiency by reducing computational requirements, although at the expense of potential loss representativeness.

rpow-d

The last *Power of Choice* strategy variant, *rpow-d*, aims to address identified weaknesses of the base strategy. It eliminates the initial phase, and selected clients return their cumulative averaged loss from local iterations when transmitting local models to the server. The server uses the most recently received loss value from each client as a proxy for the loss in client selection. For yet-to-be-selected clients, the most recent loss value is set to ∞.

5.2. Resource-aware workload allocation

To address the challenge outlined in Section 2 of substantial resource variability across devices in federated learning we present four *Global Update Optimizers* as extensions of strategies to adjust workload allocation at each client's round initiation. Expecting uniform workloads or consistent performance across devices is unrealistic, thus we aim to assign each client workload proportional to its capabilities.

Static optimizer

The *Static Optimizer* sets *epochs*, *batch_size*, and *fraction_samples* statically from configuration file parameters, treating all devices uniformly. This basic implementation, lacking consideration for device-specific features and dynamic behavior, is designated as the baseline for comparison with more sophisticated techniques.

Uniform optimizer

The *Uniform Optimizer* assigns *epochs*, *batch_size*, and *fraction_samples* to devices by drawing from uniform distributions within specified ranges, configured in YAML. For device k, the number of epochs is sampled from $[epochs\_min, epochs\_max)$. Similarly, $batch\_size_k$ and $fraction\_samples_k$ are drawn uniformly from ranges $[batch\_size\_min, batch\_size\_max)$ and $[fraction\_samples\_min, fraction\_samples\_max)$ respectively. This optimizer, aiming to emulate computational workload heterogeneity, provides a baseline for comparison against more adaptive strategies.

Round Time optimizer

The *RT (Round Time) Optimizer* configures *batch_size* and *fraction_samples* taking them from configuration file. For each client k, the number of epochs is assigned proportionally to its computational power, measured by the *iterations per second* (IPS). The device with the highest IPS is assigned the maximum epochs specified in the configuration. This approach optimally utilizes faster devices by allocating more computational load, preventing slowdowns from slower devices with proportionally reduced epochs. It exemplifies resource-aware workload allocation, dynamically adjusting to variations in clients' computational capacities.

Equal Computation Time optimizer

The *Equal Computation Time (ECT) Optimizer* configures *epochs*, *batch_size*, and *fraction_samples* based on a fixed computation time (*comp_time*) and individual client *IPS* values.
For a client k with IPS_k, local iterations are computed as

$$local\_iterations_k = comp\_time \cdot IPS_k$$

If the number of epochs is the varying parameter, $epochs_k$ is computed as:

$$epochs_k = \frac{\text{local\_iterations}_k \cdot \text{batch\_size}}{num\_samples_k \cdot \text{fraction\_samples}}$$

The optimizer allows variation in one parameter, with the others set from the configuration file. The choice of the varying parameter depends on the scenario; for instance, varying the fraction of samples risks inadequate updates for low *IPS*, increased batch size may stress limited memory, and excessive epochs can lead to

overfitting. This approach ensures equal computation time across devices, addressing resource variations.

6. Experiments

In this section, we detail the setup and execution of experiments designed to evaluate the performance of implemented federated learning techniques, as introduced in Section 5. The experiments aim to assess the impact of these strategies on the federated learning process, considering metrics such as training accuracy, convergence speed, test accuracy, and loss. We exploited the built-in Flower simulation engine to run two types of neural networks, a Multi-Layer Perceptron and a deep convolutional neural network, on image classification tasks, specifically using the MNIST and CIFAR-10 datasets. To simulate realistic non-IID data distribution among clients, the experiments utilize a data partitioning method based on the Dirichlet distribution. Parameter α controls the extent of data heterogeneity, with a smaller value of α indicating greater data heterogeneity. The simulations are executed on the CPU, with the option to switch to GPU for larger networks. Flower's efficiency in managing a large number of clients is leveraged to emulate a realistic federated environment with diverse client characteristics. The detailed setup ensures a comprehensive evaluation of techniques under various conditions, allowing for a nuanced analysis of their strengths and weaknesses. We provide specific values and configurations of experiments to enable the reproducibility of results.

In Figure 1, we present a plot depicting results from experiments, specifically comparing dynamic client selection strategies for CNN on the CIFAR10 dataset partitioned with $\alpha = 0.6$ to create an unbalanced dataset. The findings indicate the competitiveness of the dynamic strategies against the static FedAvg strategy baseline. In Figure 2, we present a plot showing results comparing resource-aware workload optimizers for MLP on the MNIST dataset partitioned with $\alpha = 0.6$. The findings indicate the competitiveness of our optimizers, dynamically considering system properties, against strategies solely optimizing accuracy disregarding such considerations.

The experiments' results allowed us to formulate

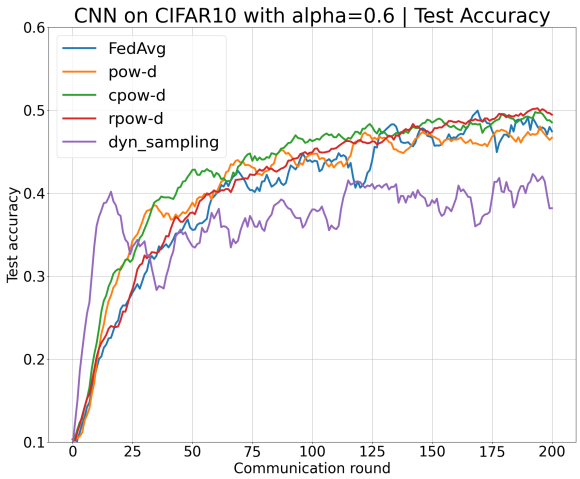

Figure 1: Dynamic selectors, CNN on CIFAR10 with *alpha*=0.6, Test accuracy.

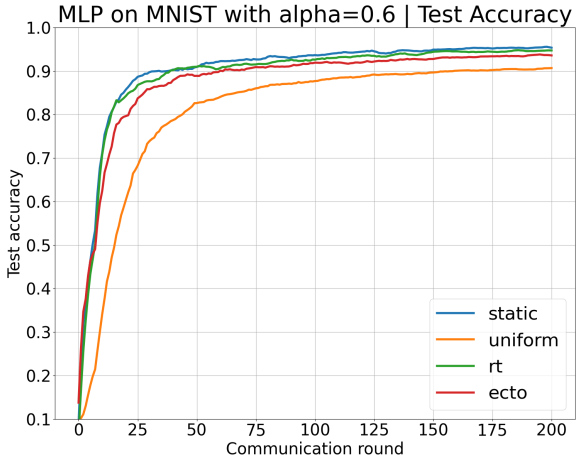

Figure 2: Workload optimizers, MLP on MNIST with *alpha*=0.6, Test accuracy.

the following considerations:

- The *Power of Choice* strategies exhibited effectiveness, particularly in unbalanced datasets, outperforming *FedAvg* in terms of convergence speed and stability.
- The *dynamic sampling* strategy, while converging quickly, displayed performance instability over time, attributed to its initial involvement of a large number of clients.
- The *static* and *RT* optimizers consistently performed well across diverse settings, with the latter's adaptive workload distribution proving to be a close and efficient competitor.
- The *uniform* optimizer, relying on a random approach to workload distribution, yielded poorer performance, emphasizing the necessity for strategies considering

client-specific characteristics in federated learning scenarios.

In summary, our experiments underscored the significance of strategic client selection and workload distribution in federated learning, particularly in heterogeneous and real-world scenarios, to achieve effective and stable model training.

7. Conclusions

This thesis proposes extensions to Flower, a highly promising framework for advancing federated learning research.

The platform offers several advantages:

- A comprehensive set of classes and interfaces that allow easy extension and integration of new strategies and techniques.
- Very well-maintained documentation and an active community to support the development of the project.
- An easy-to-use simulation engine that allows replication of federated learning environments close to reality, enabling easy reproducibility of experiments and prototyping.

Building on this platform, our objective was to address existing gaps in federated learning research, specifically focusing on crucial open issues such as dynamic client selection during training and workload allocation considering client properties and resources.

We presented four state-of-the-art strategies for dynamic client selection, extended Flower to accommodate them, and compared them against the *FedAvg* baseline.

Subsequently, we proposed four strategies for resource-aware workload allocation, extended Flower to support them and integrated them with the dynamic client selection techniques.

We finally experimented with the implemented techniques by utilizing the built-in Flower simulator. We simulated a federated learning setting with 100 clients over 200 rounds and compared the implemented techniques.

Our experiments revealed the competitiveness of the proposed strategies with state-of-the-art techniques, showcasing superior performance in certain settings, such as heterogeneous clients.

Part of our work is currently in the process of being merged into the official repository of Flower, which boasts over 3300 stars on GitHub. This contribution positions us as part of the next generation of research in federated learning.

7.1. Future work

Even though the work considered multiple facets of federated learning, it could be extended in various directions:

- Further investigate dynamic selection and resource-aware workload allocation by proposing innovative strategies that take into account multiple client properties, such as battery life, signal level, or network speed.
- Experiment with other tasks, such as natural language processing with advanced models like GPT and LLaMA, fueling research interest in federated learning for such complex models.
- Experiment with a real federated environment with multiple actual devices. These experiments may reveal nuances not evident in a simulated environment, such as clients slowing down training due to lower computational power.

References

[1] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly federated learning research framework. *CoRR*, abs/2007.14390, 2020.

[2] Shaoxiong Ji et al. Dynamic sampling and selective masking for communication-efficient federated learning. *CoRR*, abs/2003.09603, 2020.

[3] Yae Jee Cho et al. Towards understanding biased client selection in federated learning. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 10351–10375. PMLR, 28–30 Mar 2022.

[4] Brendan McMahan et al. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.