# Chapter 1

# Introduction

# Chapter 2

# Federated Learning

## 2.1 Introduction

In the rapidly evolving landscape of artificial intelligence and machine learning, Federated Learning (FL) has emerged as a paradigm that addresses key challenges related to privacy, data security, and decentralized computing. Federated Learning represents a novel approach to model training, allowing machine learning models to be trained collaboratively across multiple decentralized devices or servers without exchanging raw data [3].

Unlike traditional centralized approaches, where data is collected and processed in a central server, FL enables training on local devices following a scheme of decentralized model training. This decentralization ensures that data remains on the device, granting a certain degree of privacy. In a centralized setting, the trained model is updated based on the complete dataset, which is stored in a unique server. In the federated setting, data is distributed across local devices (parties) and training happens locally. This can be problematic, since the source of the data differ, the data can differ in various ways: unbalanced datasets, different distributions, etc. This is one of the main challenges of FL: **non-IID data**, since this will negatively affect the performance of the model [2], [8], [1].

Horizontal Federated Learning (HFL) and Vertical Federated learning (VFL) are two variations of the federated learning paradigm that differ in how they distribute and collaborate on data.

- **HFL:** Each party has a portion of the overall dataset, each party holds a different subset of samples but for the same features.
- **VFL:** The data is vertically partitioned, each party has different features for the same set of samples.

[**PONER TABLAS QUE ILUSTREN HFL Y VFL**]

HFL and VFL are not mutually exclusive, in some cases a combination of both schemes may be applied. This work will focus on HFL. Also, we will only study *Cross-Silo Federated Learning (Cross-Silo FL)*, which is a variation of FL that addresses the scenario where data is distributed across different organizations, usually few parties, each maintaining control over its own data. This setting is particularly relevant in industries where different organizations need to collaborate on machine learning task, such as healthcare (hospitals collaborating on medical research), finance (banks collaborating on fraud detection), epidemiological studies (international public health agencies studying disease spread), smart cities (urban planning authorities collaborating on public services optimization), etc. Ensuring interoperability between different silos is a huge challenge, since there needs to be a fixed standard in data format, structures and processing capabilities accross different organizations.

Another flavour of FL is *Cross device FL*, which was the original topology proposed [3]. In this type of setting, the scale of the parties is potentially much bigger (order of millions) since it's aimed to mobile devices, IoT, apps... where the data format and architecture is usually already defined by an organization. For example, this is the kind of FL that Google uses for training Gboard's models [7]. With *Cross device FL*, it's

needed to take into account some technical difficulties like client selection strategy [4], connection overheard, connection dropouts, device heterogeneity, etc. Some of these challenges are shared with *Cross-Silo FL*. The main differences are the scale of parties, the computing resources (mobile devices vs data centers or computers) and the partition of data: *cross device* tends to be HFL while *cross-silo* could be HFL or VFL. Since we are focusing in *cross-silo FL*, it won't be studied the implications of client selection (since we are assuming the parties are few and well-defined), the connection overheard or dropouts. The focus of this work will be mainly the statistical heterogeneity challenge, since this work serves as the final project for obtaining a MSc in Statistics.

We will begin by studying the FedAvg algorithm [3], which is de facto approach for Federated Learning (FL). We will establish notation, examine some of its properties, and explore issues that arise when data is not independently identically distributed (statistical heterogeneity). Following that, various approaches that have been proposed to address this problem will be developed, and the performance of each will be analyzed across different training architectures.

## 2.2 FedAvg

Let $D = \{(\mathbf{x}, y)\}$ be the global dataset[1] and $D^i \subset D$ the $i$-th party's local dataset, $i = 1, ..., N$. Let $\omega_g^t$ and $\omega_i^t$ be the global model and the local model of the $i$-th party in round $t \in \{1, ..., T\}$, respectively. Since we are working in a *Cross-silo FL* setting, the main differences between a federated optimization scheme and a distributed one would be the unbalanced data and non-IID data.
In a general FL optimization framework, we want to minimize the objective function:

$$F(\omega) = \mathbb{E}_{i\sim D}\Big[F_i(\omega_g)\Big], \qquad F_i(\omega_g) = \mathbb{E}_{z\sim D^i}\Big[l_i(\omega_g, z)\Big] \tag{2.1}$$

Here, $l_i(\omega_g, z)$ represents the local loss function of client i (with local dataset $D^i$). As stated in [5], the objective function in 2.1 can take the form of an empirical risk minimization objective function:

$$F(\omega_g) = \sum_{i=1}^{N} \alpha_i F_i(\omega_g) \text{ where } F_i(\omega_g) = \frac{1}{|D^i|} \sum_{z\in D^i} l_i(\omega_g, z) \text{ and } \sum_{i=1}^{N} \alpha_i = 1 \tag{2.2}$$

If $\alpha_i = \frac{|D^i|}{\sum_{i=1}^{N} |D^i|}$, the objective function in 2.2 would be the empirical risk minimization objective function of $D = \cup_i D^i$. We recall that 2.1 is an optimization problem that could be solved using Stochastic Descent: $\omega_g^{t+1} = \omega_g^t - \eta_t \nabla F(\omega_g^t)$ where $\eta_t$ is the learning rate at time $t$. In practice, we will use an unbiased estimator of the gradient of the local loss $g_i(\omega_g^t)$ such that $\mathbb{E}_{z\sim D^i}\Big[g_i(\omega_g^t)\Big] = \nabla F_i(\omega_g^t)$ using Stochastic Gradient Descent (SGD).

In [3], it was introduced the **FederatedAveraging** algorithm (FedAvg).

[1] Here, the global dataset is the union of the different local datasets, $D = \cup_{i=1}^{N} D^i$ . In practical cases, there is no such dataset in order to ensure data privacy. However, we will consider it to conduct a performance study of the various algorithms.

**Algorithm 1** FedAvg

**Input:** local datasets $D^i \; \forall i \in \{1, \dots, N\}$, number of parties $N$, number of communication rounds $T$, number of local epochs $E$, learning rate $\eta$ for SGD, local mini-batch size $B$.
**Output:** global model $\omega_g^T$.

**procedure** SERVER EXECUTION
  Initialize $\omega_g^0$
  **for** round $t = 1, \dots, T$ **do**
    $S_t$ (Selection of clients)
    **for** client $k \in S_t$ **in parallel do**
      $\omega_k^{t+1} \leftarrow ClientUpdate(k, \omega_g^t)$
    $\omega_g^{t+1} \leftarrow \sum_{k \in S_t} \frac{|D^k|}{\sum_{k \in S_t} |D^k|} \omega_k^{t+1}$
**procedure** $ClientUpdate(k, \omega_g^t)$
  $\omega_k^t \leftarrow \omega_g^t$
  $\mathcal{B} \leftarrow$ Batches of $D^k$ of size $B$
  **for** local epoch $i = 1, \dots, E$ **do**
    **for** batch $\mathbf{b} \in \mathcal{B}$ **do**
      $\omega_k^t \leftarrow \omega_k^t - \eta \nabla l(\omega_k^t; \mathbf{b})$
  return $\omega_k^t$ to the server.

As we see, *FedAvg* took into account a client selection strategy. Since we are focusing in a framework where the clients are limited, we will not consider any subset of clients each round: all the clients will participate.
The algorithm is quite simple, the server is simply averaging the local models' weights, with a constant parameter determining the contribution of each client given the size of the local dataset.
Each client performs multiple local epochs, which reduces the communication rounds. However, these local updates can lead to a worse performance since the local objective functions differ from each other. This makes the algorithm not robust against non-IID data. There are several works that try to tackle this problem and a lot of FedAvg variants have been proposed in order to mitigate the divergence between the local updates. We will review some of the more relevant of them, starting with *FedProx*.

## 2.3 FedProx

In [2], a generalization of *FedAvg* was proposed where the local function to minimize $F_k(\omega_k)$ was modified adding a proximal term:

$$\min_{\omega_k} h_k(\omega_k, \omega_g^t) = F_k(\omega_k) + \frac{\mu}{2} \|\omega_k - \omega_g^t\|^2, \qquad \mu \geq 0 \tag{2.3}$$

The basic idea, is to restrict the local updates to be closer to the actual global model. In the original proposal, it was stated that a constant number of local epochs per round each time could be not feasible because of the device / system heterogeneity. Since this work is not focused on system heterogeneity, it will not be mentioned the theoretical convergence results.

*FedAvg* is a particular case when $\mu = 0$, a fixed number of local epochs $E$ is set and all the local solvers are SGD. As $\mu$ increases, the function becomes more restrictive meaning that it will take longer to converge. This algorithm doesn't restrict us to any local solver, so it's not necessary to compute the estimation of the gradient using SGD. When the data across clients is IID, a positive $\mu$ could decelerate the convergence, some heuristics could be applied such as decreasing $\mu$ as long as the loss functions continues to decrease. Following the experiments in [2], the possible values of $\mu$ that will be used in this work are selected from the finite set $\{0.001, 0.01, 0.1, 1\}$

## 2.4 FedNova

This algorithm was first introduced in [6], in order to study its main contributions we are changing the original notation of *FedAvg*. We have seen that the client updates its model at round $t$ starting with a shared, global model $\omega_g^t$. Then, it applies SGD with its local data for a fixed number of epochs E. We recall that, for each epoch, and for each batch $\mathbf{b} = \{\mathbf{x}, y\} \in \mathcal{B} \subset D^i$, the i-th client performs $\omega_i^t = \omega_i^t - \eta \nabla l_i(\omega_i^t, \mathbf{b})$. After the round, the client will have a locally updated model $\omega_i^{t+1}$. After all the clients have updated their models, it aggregates them following the formula: $\omega_g^{t+1} = \sum_{k \in S_t} \frac{|D^k|}{\sum_{k \in S_t} |D^k|} w_k^{t+1}$. We can now consider the difference between the global client's starting model $\omega_g^t$ and its locally updated model $\omega_i^{t+1}$, we will denote $\Delta\omega_i^t := \omega_g^t - \omega_i^{t+1}$, the difference between the global model at time $t$ and the local model of client $i$ at time $t+1$. The server would then collect $\Delta\omega_k^t \ \forall k \in \{1, \dots, N\}$ and update the global model. For example, in *FedAvg*: $\omega_g^{t+1} = \omega_g^t + \sum_{k \in S_t} \alpha_k \Delta\omega_k^t = \omega_g^t - \sum_{k \in S_t} \alpha_k \eta \sum_{j=1}^{\tau_k} g_k(\omega_k^{t,j})$ where $\alpha_k = \frac{|D^k|}{\sum_{k \in S_t} |D^k|}$, $\tau_k = \lfloor \frac{E_k |D^k|}{B} \rfloor$ with $B$ being the mini-batch size and $E_k$ the local number of epochs of client $k$, $\omega_k^{t,j}$ is the client k's model after the j-th SGD update at time $t$, $\eta$ is the learning rate (same for all clients) and $g_k$ is the stochastic gradient over a mini-batch $\mathbf{b} \in \mathcal{B}$.
As stated in [6], different $E_k$ across clients and rounds means that $FedAvg$ algorithm converges to a surrogate objective instead of $F(\omega)$ from (2.1). This algorithm considers that different parties compute a different number of local steps ($\tau_k$) because of computation constraints or different sizes of local datasets. When $\tau_k > \tau_{k'}$, client $k$ will have a major influence over the global update rule compared to client $k'$. That's why the authors proposed to normalize and scale the local updates according to the number of local steps.

**[PONER IMAGEN QUE SE VEA LAS DIRECCIONES DE LOS GRADIENTES CUANDO NO SE NORMALIZA Y CUANDO SE NORMALIZA]**

**Algorithm 2** FedNova

**Input:** local datasets $D^i \ \forall i \in \{1, \dots, N\}$, number of parties $N$, number of communication rounds $T$, number of local epochs $E$, learning rate $\eta$, local mini-batch size $B$.
**Output:** global model $\omega_g^T$.

**procedure** SERVER EXECUTION
  Initialize $\omega_g^0$
  **for** round $t = 1, \dots, T$ **do**
    $S_t$ (Selection of clients)
    **for** client $k \in S_t$ **in parallel do**
      $\Delta\omega_k^t, \tau_k \leftarrow ClientUpdate(k, \omega_g^t)$
    $n \leftarrow \sum_{k \in S_t} |D^k|$
    $\omega_g^{t+1} \leftarrow \omega_g^t - \eta \frac{\sum_{k \in S_t} |D^k| \tau_k}{n} \sum_{k \in S_t} \frac{|D^k| \Delta\omega_k^t}{\tau_k n}$
**procedure** $ClientUpdate(k, \omega_g^t)$
  $\omega_k^t \leftarrow \omega_g^t$
  $\tau_k \leftarrow 0$
  $\mathcal{B} \leftarrow$ Batches of $D^k$ of size $B$
  **for** local epoch $i = 1, \dots, E$ **do**
    **for** batch $\mathbf{b} \in \mathcal{B}$ **do**
      $\omega_k^t \leftarrow \omega_k^t - \eta \nabla l(\omega_k^t; \mathbf{b})$
      $\tau_k \leftarrow \tau_k + 1$
  $\Delta\omega_k^t \leftarrow \omega_g^t - \omega_k^t$
  return $\Delta\omega_k^t$, $\tau_k$ to the server.

## 2.5 SCAFFOLD

With data heterogeneity, *FedAvg* suffers from client-drift, since each local updates it's minimizing its own local objective. SCAFFOLD tries to solve this introducing control variables for the server $c$ and the clients $c_k$ $\forall k \in \{1, \dots, N\}$.

**Algorithm 3** SCAFFOLD

**Input:** local datasets $D^i$ $\forall i \in \{1, \dots, N\}$, number of parties $N$, number of communication rounds $T$, number of local epochs $E$, learning rate $\eta$, local mini-batch size $B$.
**Output:** global model $\omega_g^T$.

1: **procedure** SERVER EXECUTION
2: Initialize $\omega_g^0$
3: $c^t \leftarrow \mathbf{0}$, $c_k \leftarrow \mathbf{0}$
4: **for** round $t = 1, \dots, T$ **do**
5: $S_t$ (Selection of clients)
6: **for** client $k \in S_t$ **in parallel do**
7: $\Delta\omega_k^t, \Delta c_k \leftarrow ClientUpdate(k, \omega_g^t, c^t)$
8: $n \leftarrow \sum_{k \in S_t} |D^k|$
9: $\omega_g^{t+1} \leftarrow \omega_g^t - \eta \sum_{k \in S_t} \frac{|D^k|}{n} \Delta\omega_k^t$
10: $c^{t+1} \leftarrow c^t + \frac{1}{N} \sum_{k \in S_t} \Delta c_k$
11: **procedure** $ClientUpdate(k, \omega_g^t, c_k)$
12: $\omega_k^t \leftarrow \omega_g^t$
13: $\tau_k \leftarrow 0$
14: $\mathcal{B} \leftarrow$ Batches of $D^k$ of size $B$
15: **for** local epoch $i = 1, \dots, E$ **do**
16: **for** batch $\mathbf{b} \in \mathcal{B}$ **do**
17: $\omega_k^t \leftarrow \omega_k^t - \eta(\nabla l(\omega_k^t; \mathbf{b}) - c_k + c)$
18: $\tau_k \leftarrow \tau_k + 1$
19: $\Delta\omega_k^t \leftarrow \omega_g^t - \omega_k^t$
20: $c_k^+ \leftarrow$ (i) $\nabla l(\omega_g^t)$ or (ii) $c_k - c + \frac{1}{\tau_k \eta}(\omega_g^t - \omega_k^t)$
21: $\Delta c \leftarrow c_k^+ - c_k$
22: $c_k = c_i^+$
23: return $\Delta\omega_k^t$, $\Delta c$ to the server.

From the SCAFFOLD's algorithm, it can be noted that the clients maintain the state of $c_k$ and the server maintain $c$, which are initialized at 0. In comparison with the previous algorithms, here the local update formula is: $\omega_k^t = \omega_k^t - \eta(\nabla l(\omega_k^t, \mathbf{b}) - c_k + c)$. Intuitively, the local gradient $\nabla l(\omega_k^t, \mathbf{b})$ moves towards to the local optimum for client $k$, the correction $c - c_k$ ensures that the update change its direction towards the global optimum. As we can see, in line 20, there are two options for computing $c_i^+$: $\nabla l(\omega_g^t)$ or $c_k - c + \frac{1}{\tau_k \eta}(\omega_g^t - \omega_k^t)$.

# Bibliography


[1] Qinbin Li et al. *Federated Learning on Non-IID Data Silos: An Experimental Study*. Oct. 2021. arXiv: 2102.02079 [cs]. (Visited on 01/12/2024).

[2] Tian Li et al. *Federated Optimization in Heterogeneous Networks*. Apr. 2020. arXiv: 1812.06127 [cs, stat]. (Visited on 12/16/2023).

[3] H. Brendan McMahan et al. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. Jan. 2023. arXiv: 1602.05629 [cs]. (Visited on 01/12/2024).

[4] Carl Smestad and Jingyue Li. "A Systematic Literature Review on Client Selection in Federated Learning". In: *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*. June 2023, pp. 2–11. DOI: 10.1145/3593434.3593438. arXiv: 2306.04862 [cs]. (Visited on 05/11/2024).

[5] Jianyu Wang et al. *A Field Guide to Federated Optimization*. July 2021. arXiv: 2107.06917 [cs]. (Visited on 05/14/2024).

[6] Jianyu Wang et al. *Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization*. July 2020. arXiv: 2007.07481 [cs, stat]. (Visited on 05/15/2024).

[7] Yuanbo Zhang et al. *Private Federated Learning in Gboard*. June 2023. arXiv: 2306.14793 [cs]. (Visited on 05/11/2024).

[8] Yue Zhao et al. "Federated Learning with Non-IID Data". In: (2018). DOI: 10.48550/arXiv.1806.00582. arXiv: 1806.00582 [cs, stat]. (Visited on 01/12/2024).