Enabling Federated Learning across the Computing Continuum: Systems, Challenges and Future Directions

Cédric Prigent, Alexandru Costan, Gabriel Antoniu, Loïc Cudennec

Enabling Federated Learning across the Computing Continuum: Systems, Challenges and Future Directions

Cédric Prigent[a], Alexandru Costan[a], Gabriel Antoniu[a] and Loïc Cudennec[b]

[a] University of Rennes, Inria, CNRS, IRISA, Rennes, France
[b] DGA Maîtrise de l'Information, Rennes, France

ARTICLE INFO

Keywords:
Computing Continuum
Federated Learning
Edge Computing
Personalization
Privacy and Security
Frameworks

ABSTRACT

In recent years, as the boundaries of computing have expanded with the emergence of the Internet of Things (IoT) and its increasing number of devices continuously producing flows of data, it has become paramount to boost speed and to reduce latency. Recent approaches to this growing complexity and data deluge aim to integrate seamlessly and securely diverse computing tiers and data environments, spanning from core cloud to edge - the Computing Continuum (or Edge-to-Cloud Continuum). Typically, the cloud is used for resource-intensive computations while the edge for low-latency tasks.

This provides an opportunity to run complex AI-enabled applications across multiple tiers specifically facilitating the training of Machine Learning (ML) models at the "edge" of the Internet (*i.e.,* beyond centralized computing facilities such as cloud datacenters). Federated Learning (FL) represents a novel ML paradigm for collaborative training, capitalizing on processing capabilities at the edge for training purposes while addressing privacy concerns. A set of clients (*i.e.,* edge devices) collaboratively train a shared model under the supervision of a centralized server without exchanging personal data. However, several challenges arise from the decentralized nature of FL in the Computing Continuum context: statistical heterogeneity (data drift between parties), system heterogeneity (due to the nature of the environment), volatility (*e.g.*, client dropouts), security threats and, persistently, privacy (although no personal data is transmitted, the shared model updates include information about data used for training).

As opposed to previous studies dedicated to federated learning (typically on homogeneous, edge-based infrastructures), this survey aims to present a systematic overview of the existing literature addressing how state-of-the-art Federated Learning (FL) systems contend with the challenges previously outlined within the edge-to-cloud Computing Continuum, in particular heterogeneity, volatility and large-scale distribution. We analyze representative tools for implementing, monitoring, configuring and deploying such systems. We highlight significant efforts made to overcome statistical heterogeneity and security problems in FL. We specifically analyze the quality of the experimental evaluation done for existing systems and the relevant benchmarks. Finally, we discuss some open issues and future directions (*e.g.*, lack of experiments in realistic environments) to support the broader adoption of FL across the continuum and to eventually fulfill the vision of the AI and edge computing convergence - the edge intelligence.

1. Introduction

The number of IoT devices generating more and more data at the edge has greatly increased in recent years. Many organizations see there an opportunity to massively store and process personal data for training machine learning (ML) models. However, data protection regulations such as GDPR in EU or CCPA in US set many constraints and directives to follow in order to process those data (*e.g.*, data minimization principles, security measures). Cloud-based solutions that have been used for years to train ML models in a centralized manner are now facing numerous constraints due to these new data protection regulations.

Alternative solutions advocate for decentralization by moving the AI closer to or at the edge of the network, where the data is produced. This shift towards *edge intelligence* [1] is now accelerated by the emergence of novel systems enabling the concurrent execution across multiple computing tiers (IoT, edge, fog, clouds). Commonly referred to as the **Edge-to-Cloud Continuum** or the **Computing Continuum** (CC), this approach leverages the strengths of each computing tier (heavy computations on the clouds, low-latency processing at the edge) while circumventing their respective limitations through specific optimisations and new decentralised solutions for historically centralised problems (*e.g.,* learning).

Federated Learning (FL) is such an alternative, enabling ML in decentralized environments to address privacy concerns and satisfy regulatory compliance. In FL, a set of clients collaboratively train a ML model by periodically performing local training and sending updated versions of the model to a central server. FL takes advantage of the recent improvements in computational resources at the edge to perform on-device training of the model without communicating raw user data to the server and instead exchanges model weights as illustrated in Figure 1. FL brings significant advantages over centralized settings.

Improving privacy: In a centralized setting, data are sent to remote servers and then processed using cloud or HPC

cedric.prigent@inria.fr (C. Prigent); alexandru.costan@inria.fr (A. Costan); gabriel.antoniu@inria.fr (G. Antoniu); loic.cudennec@intradef.gouv.fr (L. Cudennec)

ORCID(s): 0000-0002-1836-7965 (C. Prigent); 0000-0003-3111-6308 (A. Costan); 0000-0001-6525-3736 (G. Antoniu); 0000-0002-6476-4574 (L. Cudennec)

resources. This can lead to privacy risks, and therefore, individuals may not want their data to be collected. In FL, raw personal data is not exchanged between parties, instead it is processed locally on user devices.

Learning from more diverse data: As FL takes advantage of decentralized processing from edge devices to train ML models, it benefits from more data diversity and is more representative of the current trends in the different populations of devices. Traditional centralized ML is limited either by privacy concerns (*e.g.*, collaboration between hospitals) or scaling issues (*e.g.*, with the increasing volumes of data to process).

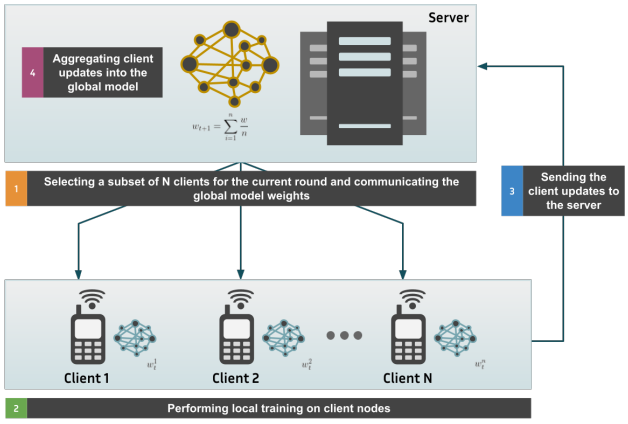
Reducing the pressure on the servers: With the exploding volume of data continuously generated at the edge, the required bandwidth, storage, and computing resources needed to upload and process data in real-time can become a bottleneck for cloud-based solutions. FL takes advantage of edge resources to directly process data at their generation sites, reducing the pressure on the servers.

At the same time, FL faces several challenges due to its decentralized nature: *system heterogeneity* and *statistical heterogeneity*, as well as *ensuring client participation* and *security*. The heterogeneous and volatile nature of the resources across the Computing Continuum complicate these issues even further. This article proposes a structured analysis of the literature on FL focusing on the existing solutions to address these challenges. The main contributions of this survey are:

- An overview of the **key concepts in FL**, characterizing the nature of the collaborative training (*e.g.*, scale and nature of the data partitioning).
- A **survey** of the FL literature resulting in a **taxonomy of existing systems and tools** built from the perspective of how they address the challenges of supporting FL in the Computing Continuum.
- A set of discussions providing a **critical perspective** on existing works and identifying **promising directions to cope with the limitations**.
- A set of **open challenges** that need further investigation to better support the development of FL systems in the context of the Computing Continuum, such as: addressing the concept drift; exploring unsupervised approaches; enabling seamless composability of FL components and reconfigurability of the system; as well as sustainability in order to support more heterogeneity in FL.

The remainder of this article is organized as follows. In Section 2, we discuss previous surveys on FL and motivate the need for a new one. Section 3 describes the methodology used for conducting this survey. Section 4 gives background about FL and the CC, and highlights the current challenges for the adoption of FL across the CC. Section 5 provides a detailed overview of FL systems with regard to the challenges they want to address. Section 6 lists frameworks

Figure 1: Federated Learning Process



that can be used to implement FL systems and approaches for the configuration of the application. Section 7 provides more information about model validation as well as model management and deployment tools. Section 8 is dedicated to open issues and research opportunities for the broader adoption of FL across the CC.

2. New Contributions of this Study

Several studies were conducted to assess advances in Federated Learning. In this section we position our survey with respect to these prior works.

Related studies. The basic concepts and the architecture of FL were first introduced in [2, 3, 4], and were illustrated with several use-cases (*e.g.*, mobile devices, industrial engineering, healthcare). Two extensive reviews about FL at the Edge of the continuum are proposed in [5, 6]. [5] reviews FL studies from two perspectives: *enabling FL at the Edge* and *Edge platforms in support of FL*. Authors of [6] present protocols, architectures and specific issues of existing approaches for FL at the Edge (*e.g.*, heterogeneity management, security). The underlying implementation challenges for mobile edge computing are presented in [7]. Authors of [8] focus on the problem of Non-IID data in FL (*i.e.*, data drift among clients). In this work, different types of data skew are described and several approaches to handle them are proposed. In [9], the authors present challenges related to Vertical FL (*i.e.*, training from vertically partitioned data) and discuss possible solutions. Several approaches to optimization and implementation of FL in resource-constrained IoT are presented in [10]. In [11] and [12], a functional architecture and a collection of architectural patterns for FL systems are discussed. Authors of [13] propose a definition of FL systems and classify them according to multiple criteria (*e.g.*, data partitioning, model implementation, communication architecture). A taxonomy of threat models and major attacks in FL is provided by [14]. Finally, some studies focus on the application layer, like [15, 16] presenting specific FL designs, privacy and security measures for smart healthcare.

While all these studies address important issues of Federated Learning in homogeneous settings (*i.e.,* only at the edge), none of them considered it from the Computing Continuum perspective, that is, presenting the current approaches and the challenges of applying FL in highly distributed, heterogeneous and volatile environments. In [17], authors propose a taxonomy of heterogeneous FL environments and present few directions to address it such as personalized FL and transfer learning, however they do not directly target highly distributed and volatile environments. Decentralized FL approaches are reviewed in [18] but remain specific to peer-to-peer FL. [19] reviews distributed learning approaches (FL and split learning) in the IoT-Edge-Cloud Continuum, yet specially targeting privacy and security. Other studies were proposed with a focus on the Computing Continuum [20, 21], however they do not deal with FL. We aim to fill this gap and provide further background, challenges and perspectives of FL in the context of the Computing Continuum.

Originality of this study. Our goal is to provide an overview of the main challenges and the latest progress towards the adoption of FL across the continuum. This survey brings several new contributions compared to previous studies:

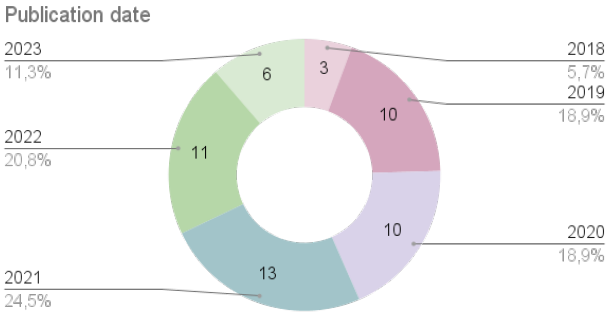
- We highlight a set of **challenges of the FL support across the Computing Continuum**.
- We propose a **taxonomy of FL systems** to solve these challenges.
- We provide an extended overview of **frameworks and benchmarks** for implementing FL systems with respect to the **supported settings, training modes and ML frameworks** they natively interface with.
- We are **the first to systematically present** details of the **experimental evaluation** such as the scale of experiments, datasets and platforms used to validate those FL systems, in an effort to support **reproducibility** for further analysis and optimisation within the community.
- We present a set of **deployment and model management tools** to facilitate the **supervision of FL systems** in this highly distributed environment.

3. Survey Methodology

To find articles that address the different challenges of supporting FL in the CC, we used several Digital Libraries: Google Scholar, IEEE Xplore, arXiv and ACM Digital Library and searched for the following terms:

- (Volatile OR Heterogeneous OR Personalized OR Privacy-preserving OR Decentralized) AND Federated Learning
- Federated Learning AND (Computing Continuum OR Client Selection)

Figure 2: Publication date of selected articles



- Federated Learning AND (Framework OR Benchmark OR Hyperparameter Optimization OR Evaluation)

Then we iteratively discarded articles with non-relevant titles and in a second stage kept the most relevant ones after reading their abstract, introduction and conclusion. Overall, in this survey, we selected and reviewed 53 articles published between 2018 and 2023 as illustrated on Figure 2. We went further by reviewing 21 implemented FL tools (*i.e.*, frameworks and benchmarks) hosted on online repositories such as GitHub.

4. Main Concepts and Current Challenges

In this section, we present some background about FL before introducing the main challenges for its seamless adoption across the Computing Continuum.

4.1. Federated Learning Overview

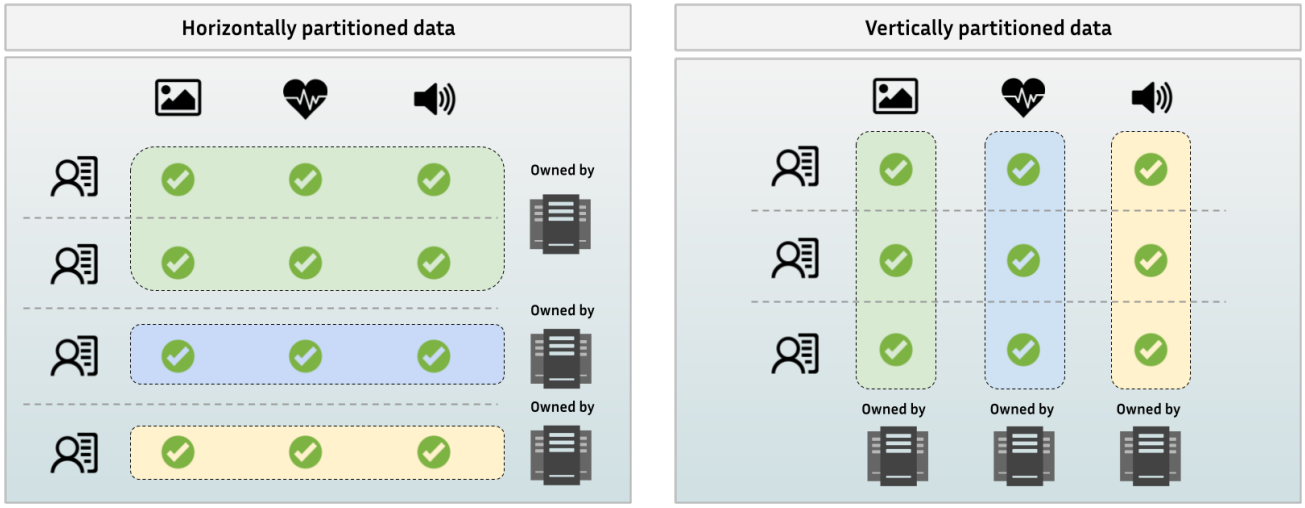
Federated Learning (FL) is a distributed paradigm for collaborative training of machine learning (ML) models. FL is typically deployed in a setup composed of an orchestrating server and a set of distributed clients at the Edge. These clients perform local training of the model over their local datasets. The server coordinates the training phase by periodically selecting subsets of clients. It then aggregates the resulting local model updates and performs a weighted averaging to build the new global model, before starting a new training round as depicted in Algorithm 1 [22].

4.2. Federated Learning Settings

FL can be classified according to several setups: *cross-device* and *cross-silo* FL are defined by the scale and nature of the clients used in the federation while *vertical* and *horizontal* FL are defined by the partitioning scheme of the training data.

In **cross-device FL**, hundreds to millions of devices with very heterogeneous power and network accessibility are used to train a model. Those devices usually have limited battery life, computational capabilities and unstable networks. Smartphones, sensors and IoMT (Internet of Medical Things) are common devices in cross-device FL. One

Figure 3: Data Partitioning Schemes



Algorithm 1 FedAvg

Input: *K is the total number of clients, C is the fraction of client to sample in each round, B is the local minibatch size, R is the total number of rounds, E is the number of local epochs; η is the learning rate and P_k refers to the partition of client k.*

Function `Server(`K, C`)`
- $w \leftarrow$ (initialize model weights)
- **foreach** *round* $\in$ `range` *(1, R)* **do**
  - $m \leftarrow max(C \cdot K, 1)$
  - $S_t \leftarrow$ (random set of m clients)
  - **foreach** *client* $k \in S_t$ **do**
    - $w_k, n_k \leftarrow$ `ClientUpdate(`k, w`)`
  - **end**
  - $n \leftarrow \sum_{i=1}^{k} n_k$
  - $w \leftarrow \sum_{i=1}^{k} \frac{n_k}{n} w_k$
- **end**
- **return** w

Function `ClientUpdate(`k, w`)`
- $batches \leftarrow$ (split P_k into batches of size B)
- **foreach** *local epoch* $\in$ `range`$(1, E)$ **do**
  - **foreach** *batch* $\in$ *batches* **do**
    - $w \leftarrow w - \eta \nabla loss(w; batch)$
  - **end**
- **end**
- **return** w, $|P_k|$

example illustrating well this setting is the mobile keyboard prediction system [23] from Google, leveraging 1.5 million Android smartphones to perform FL on local keyboard inputs. Some key challenges in cross-device FL are power-efficiency, low communication overhead, statistical heterogeneity (clients tend to have very different usage) and security (*e.g.*, IoMT).

In **cross-silo FL**, few data silos which typically come with strong computational power and stable networks are used to train a model. Those data silos are usually owned by big institutions such as data centers from hospitals or banks. Power-efficiency and low communication overhead remain important challenges in cross-silo FL, but they are less critical than in cross-device FL due to clients with strong computational power and network accessibility. In turn, privacy is a key challenge due to the nature of the data that is more likely to be sensitive [24] (*e.g.*, patient records, bank statements).

In **horizontal FL** (HFL), data is partitioned by samples over the participating parties. In other words, parties have similar data (*i.e.,* data with the same attributes) that can directly be used to train the model. For instance, mobile devices may generate log data from a texting application to train a model using FL. This setting is illustrated on the left part of Figure 3. *This setting is easily scalable, therefore it suits very well deployments across the Computing Continuum.*

In **vertical FL** (VFL), data is partitioned by features over the participating parties. In other words, parties have data with different attributes overlapping over their IDs and, through collaboration, they can train a model by combining their data. For instance, an insurance company and a bank may have overlapping data concerning their clients. Using VFL, they can train a model with data from their overlapping clients without revealing anything private to the other party. This setting is illustrated on the right part of Figure 3. *This setting is not suited for highly distributed systems, as it is hardly scalable. Instead, it suits cross-silo scenarios with very few participants. Therefore we do not focus on this setting in this study.*

Personalized FL (PFL) refers to the problem of training specialized models to better fit clients needs. It is proposed as a solution to the problem of inconsistent performance of the model caused by client drift (*i.e.*, diverging data distribution between clients). It can, for instance, take the

Figure 4: The Computing Continuum

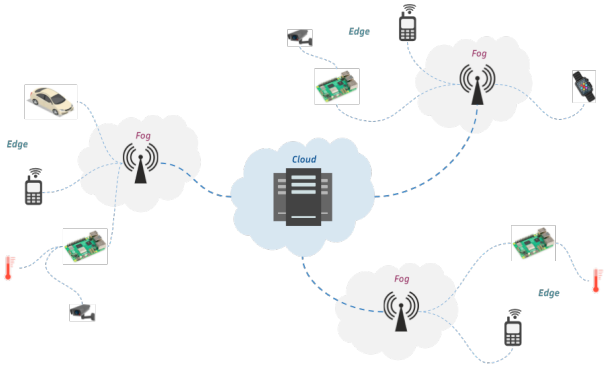
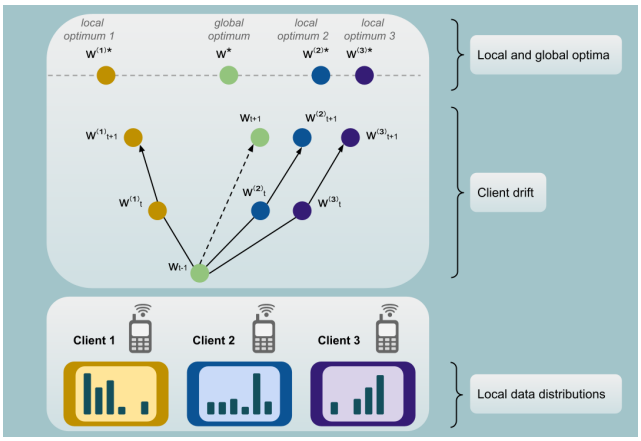


Figure 5: Client drift due to statistical heterogeneity



form of additional fine-tuning of the global model using local data to specialize it to a client's data distribution, or training specific models for clusters of similar clients.

4.3. The Computing Continuum

Edge, fog, cloud and HPC infrastructures leverage very different processing, storage and network resources. While edge devices continuously produce high volumes of data, cloud and HPC infrastructures have access to huge processing capabilities. The traditional approach for handling such volumes of data is to send them to cloud or HPC infrastructures for centralized processing. Recent improvements in processing capabilities at the edge are now enabling the processing of new data close to their generation site. By strategically distributing the processing loads between the different tiers of the continuum, we can improve latency, network usage and privacy while taking advantage of the full spectrum of available resources. With these new perspectives also come new possibilities in terms of applications (*e.g.*, smart cities, smart healthcare, digital twins etc.). We refer to this very heterogeneous and interconnected environment as the Computing Continuum [25] (Figure 4).

The majority of existing studies have considered FL to be deployed in standard environments combining Cloud resources for the server and Edge resources for the clients. Deploying FL in the CC introduces more diversity: *devices can be spread over larger geographical areas, subject to varying degrees of volatility, with different computing resources (e.g., Fog, HPC etc.) and producing different types of data*. Additionally, the recent convergence of networking and Cloud-Edge computing [26] has lead to the emergence of new network topologies such as *hierarchical* and *peer-to-peer* networks. Overall, this requires the elaboration of communication-efficient protocols as well as innovative mechanisms to ensure the successful execution of the training phase, as discussed in the next subsection.

4.4. Challenges of Supporting FL across the Computing Continuum

While Federated Learning is a promising candidate for applying ML workflows over highly distributed resources, the complex and heterogeneous nature of the continuum brings some additional challenges, that we discuss here.

1. ***Statistical Heterogeneity***. In decentralized settings, the data distribution among parties is usually Non-IID (non independent and identically distributed). Clients with diverging data distributions will likely converge to different optima and the naive strategy of averaging the local models to update the central model will not necessarily converge to the global optimum. This problem is called *client drift*, and its impact in a FL system is illustrated in Figure 5.
2. ***Volatile Environments and System Heterogeneity***. In the common FL setup, a massive number of heterogeneous devices is used. The different computing resources (*e.g.*, CPU, memory, storage), with various autonomy (*e.g.,* battery life) and volatile network accessibility may have a strong impact on the stability of the learning process.
3. ***Robustness of the System***. FL brings a privacy benefit over centralized settings by directly processing data at the edge. However, this edge training still involves exchanging model updates between parties, which may contain sensitive information about individuals. Additionally, all participating peers have full access to the model weights and therefore any malicious client has the ability to manipulate the model with regard to a malicious objective. Without appropriate privacy and security measures, the system is vulnerable to model inversion and model poisoning attacks.
4. ***Spatial Diversity***. Peers across the CC may be deployed over widely distributed geographical areas. Direct communications from the Edge to the Cloud in a single-hop manner may result in high overhead due to long transmission delays (*e.g.*, high latency, packet re-transmission). Exploring alternative network topology becomes necessary to ensure communication-efficiency.

In the following sections, we present some of the current approaches and tools proposed to address these challenges as well as the main research opportunities in this direction.

5. Current Approaches towards a Seamless Integration of FL in the CC

The heterogeneous and volatile nature of devices composing some tiers of the continuum (*e.g.,* edge, fog) has various impacts on FL: inefficient training phase, high latency, risks of security breaches, low participation rate. This calls to rethink the design of distributed applications running across the CC. In this section we provide an overview of the mechanisms applied in this direction in FL. A summary of these approaches is presented in Table 1.

5.1. Statistical Heterogeneity

The statistical heterogeneity problem in FL can be addressed through two distinct approaches [71]. The first approach consists in training one centralized model and through specific mechanisms attempting to mitigate the impact of Non-IID data in order to reach the global optimum (*i.e.*, reach a consensus among all participating peers). The second approach consists in training personalized models for each FL client or group of clients with similar data distributions to better suit each of their local needs and converge to their local optima.

5.1.1. Training One Centralized Model

*The main goal when targeting the improvement of the global model accuracy (*i.e. *model stored at the server side) is to generalize the global data distribution. Non-IID data across clients complicate generalization and improving performance of the model implies reducing the impact of Non-IID data through different mechanisms.*

A large part of previous research on FL has focused on reducing the impact of Non-IID data. FedProx [27] adds a proximal term to the local sub-problem to limit the impact of heterogeneous clients and diverging data. In [28], authors explore different server optimizers (*i.e.*, modifying the aggregation rules to update the global model) to improve the convergence of the model. FedAdagrad, FedYogi and FedAdam strategies are introduced which specifically use different optimizers at the server level. FedBN [29] addresses the problem of feature shift Non-IID data (as opposed to label distribution skew), which may occur with clients from different environments or clients using different types of sensors. The authors use local batch normalization before averaging models to reduce the impact of feature shift. FedCurv [30] focuses on building lifelong learning FL and aims to overcome catastrophic forgetting in the FL setting. The authors introduce an algorithm adding a penalty term to the loss function reacting to change in important parameters for the initial task while learning a second task. SCAFFOLD [31] attempts to solve slow convergence due to client-drift. For this, it estimates the update direction for the global model and local models and uses the difference for correction. FedNova [32] uses normalized averaging to eliminate objective inconsistency due to heterogeneous local updates. FedMA [33] constructs a shared model by matching and averaging hidden elements of the model with similar feature extraction signatures.

5.1.2. Training Personalized Models

To better fit the client needs, one direction is to provide them with personalized models. This personalization approach is motivated by the generalization problem in FL: given highly heterogeneous data, training a common model for all the users cannot suit everyone's needs. These algorithms are referred to as personalized FL (PFL). PFL is particularly relevant in the CC where the statistical heterogeneity among clients can be very high. PFL approaches can be divided in two groups: (1) user-level personalization (*i.e.*, each client has its own personalized model), and (2) cluster personalization (*i.e.*, clients are grouped into clusters for personalized FL training based on their data distributions).

User-level personalization. Several works have explored user-level personalization in FL. In [34], authors suggest finding an initial model that clients can easily adapt to their local data distributions by performing a few steps of gradient descent locally. In FedProto [35], the federation process is used to build shared prototypical networks and the training on each client focuses on minimizing the classification error on local data while keeping the resulting local prototypes close to the global prototypes. In [36], the authors highlight the similarities between FL and Model-Agnostic Meta-Learning (MAML) algorithms and propose a FedAvg variant which combines FedAvg and Reptile to provide personalized models through fine-tuning. pFedHN [37] (personalized Federated HyperNetworks) trains a central hypernetwork model on the server side using client descriptors (embedding vectors) to generate smaller personalized models for each client. It takes advantage of the server computing power to train a large central hypernetwork without communication overhead. In [38], APFL (Adaptive PFL) is a bi-level optimization algorithm learning a mixture of local and global models to provide a personalized solution. Authors of [39] propose Ditto, an algorithm optimizing two tasks, the global objective and local objectives. For this it uses a tunable regularization term encouraging personalized models to stay close to the global model which can be used to adjust the trade-off between fairness and robustness of the models. FedPer [40] is enabling personalization by training global base layers and keep locally personalized layers.

Cluster personalization. Another approach to achieve PFL is to divide the FL procedure into multiple clusters of similar clients that jointly trains personalized models. By clustering clients with similar data distributions, clustered FL keeps the benefit of collaborative training applied to a more specialized task.

Gradient similarity has been used to form clusters through recursive bipartitioning [41] and greedy agglomerative processes [44].

Clustering through *emprical risk minimization* has been proposed in [42, 43]. In IFCA [42] (Iterative Federated Clustering Algorithm), the server communicates K models corresponding to K clusters to each client. Clients iteratively estimate their cluster identity using local empirical loss functions. Authors argue that this approach benefits from releaving the server from the client clustering overhead.

Table 1
Federated Optimization Algorithms

Algorithm	Optimization Target					Evaluation Setting			
	Global Model	Personalized Models	Resource Utilization	Client Selection	Security	Data Type	Dataset	Scale of Experiments (number of clients)	Experimental Platform
Statistical Heterogeneity									
FedAvg [22]	-	-	-	-	-	Image	(1) (2)	> 500	Unknown
FedProx [27]	✓	✗	✗	✗	✗	Image/Text	(1) (2) (3) (4)	> 500	Simulated
FedOpt [28]	✓	✗	✗	✗	✗	Image/Text	(2) (5) (6) (7) (8)	> 500	Simulated
FedBN [29]	✓	✗	✗	✗	✗	Image	(1) (15) (16) (17) (18)	[20; 100[	Unknown
FedCurv [30]	✓	✗	✗	✗	✗	Image	(1)	[20; 100[	Simulated
SCAFFOLD [31]	✓	✗	✗	✗	✗	Image	(8)	[100; 200[	Simulated
FedNova [32]	✓	✗	✗	✗	✗	Image	(6)	< 20	Distributed
FedMA [33]	✓	✗	✓	✗	✗	Image/Text	(2) (6)	[20; 100[	Distributed
Per-FedAvg [34]	✗	✓	✗	✗	✗	Image	(1) (6)	[20; 100[	Simulated
FedProto [35]	✗	✓	✓	✗	✗	Image	(1) (4) (6)	[20; 100[	Simulated
FedAvg + Reptile [36]	✗	✓	✗	✗	✗	Image/Text	(4) (2)	> 500	Simulated
pFedHN [37]	✗	✓	✓	✗	✗	Image	(6) (7) (20)	[100; 200[	Simulated
APFL [38]	✗	✓	✗	✗	✗	Image	(1) (6) (8)	> 500	Distributed
Ditto [39]	✗	✓	✗	✗	✗	Image/Text	(4) (5) (10) (11) (21)	[200; 500[	Simulated
FedPer [40]	✗	✓	✗	✗	✗	Image	(4) (2) (24)	[20; 100[	Simulated
CFL [41]	✗	✓	✗	✗	✗	Image	(1) (6)	[20; 100[	Simulated
IFCA [42]	✗	✓	✗	✗	✗	Image	(1) (4) (6)	> 500	Simulated
FedSoft [43]	✗	✓	✗	✗	✗	Image	(8)	[100; 200[	Simulated
FLACC [44]	✗	✓	✗	✗	✗	Image	(1) (4) (6) (8)	[200; 500[	Unknown
LADD [45]	✗	✓	✗	✗	✗	Image	(25) (26) (27) (28)	> 500	Simulated
System Heterogeneity									
FedCS [46]	✗	✗	✓	✓	✗	Image	(6) (10)	> 500	Simulated
VFedCS [47]	✓	✗	✗	✓	✗	Image	(6) (8)	[100; 200[	Simulated
FedMarl [48]	✓	✗	✓	✓	✗	Image/Text	(1) (2) (6) (10)	> 500	Distributed
PruneFL [49]	✗	✗	✓	✗	✗	Image	(4) (6) (21) (22)	< 20	Distributed
FD + FedAug [50]	✓	✗	✓	✗	✗	Image	(1)	< 20	Unknown
SPATL [51]	✓	✓	✓	✗	✗	Image	(4) (6)	[100; 200[	Simulated
FetchSGD [52]	✗	✗	✓	✗	✗	Image/Text	(4) (6) (7) (29)	> 500	Unknown
AQFL [53]	✗	✗	✓	✗	✗	Image/Text	(2) (4) (22) (30)	> 500	Simulated
FedSyn [54]	✗	✗	✗	✗	✓	Image	(1) (6)	< 20	Distributed
FedKD [55]	✗	✗	✓	✗	✓	Image/Text	(6) (7) (31) (32) (33)	< 20	Unknown
FedAsync [56]	✓	✗	✓	✗	✗	Image/Text	(6) (9)	[100; 200[	Simulated
ASO-Fed [57]	✓	✗	✓	✗	✗	Image/Tabular	(10) (12) (13) (14)	[20; 100[	Simulated
Robust Systems									
FedCVAE [58]	✗	✗	✗	✗	✓	Image/Tabular	(1) (4) (11)	> 500	Unknown
PDGAN [59]	✗	✗	✗	✗	✓	Image	(1) (10)	< 20	Simulated
FedGuard [60]	✗	✗	✗	✗	✓	Image	(1)	[100; 200[	Distributed
FLDetector [61]	✗	✗	✗	✗	✓	Image	(1) (4) (6)	[200; 500[	Unknown
AttackerDetectionMicro [62]	✗	✗	✗	✗	✓	Tabular	(38) (39) (40)	[20; 100[	Unknown
Fed_BVA [63]	✗	✗	✗	✗	✓	Image	(1) (6) (7) (10)	[100; 200[	Simulated
Interconnecting peers across the CC									
Hierarchical FL [64]	✗	✗	✓	✗	✗	Image	(6)	[200; 500[	Simulated
HierFAVG [65]	✗	✗	✓	✗	✗	Image	(1) (6)	[20; 100[	Simulated
MFLCES [66]	✗	✗	✓	✓	✗	Image	(1)	> 500	Simulated
SHARE [67]	✗	✗	✓	✗	✗	Image	(1) (6)	[20; 100[	Simulated
ProxyFL [68]	✗	✗	✓	✗	✓	Image	(1) (6) (10) (34) (35)	< 20	Simulated
Dis-PFL [69]	✗	✓	✓	✗	✓	Image	(6) (7) (23)	[100; 200[	Simulated
FedMes [70]	✗	✗	✓	✗	✗	Image	(1) (6) (10)	[20; 100[	Simulated

(1) MNIST, (2) Shakespeare, (3) Sent140, (4) FEMNIST, (5) Stack Overflow, (6) CIFAR-10, (7) CIFAR-100, (8) EMNIST, (9) WikiText-2, (10) Fashion-MNIST, (11) Vehicle, (12) FitRec, (13) Air Quality, (14) ExtraSensory, (15) SVHN, (16) USPS, (17) SynthDigits, (18) MNIST-M, (19) STL-10, (20) Omniglot, (21) CelebA, (22) ImageNet-100, (23) Tiny-ImageNet, (24) FLICKR-AES, (25) GTA-5, (26) Cityscapes, (27) CrossCity, (28) Mapillary Vistas (29) PersonaChat (30) Reddit (31) Chest X-Ray (32) AG News (33) SST2 (34) Kvasir (35) Camelyon-17 (38) Adult Income (39) 120 years of Olympic history (40) Bank Marketing

However, while decreasing the computation burden for the server, it increases the communication overhead by sending multiple copies of the model to each client. FedSoft [43] follows a similar approach, but rather than assigning a single cluster per client, allows the assignment of several soft clusters for each client.

Finally, knowledge extraction has been used to extract meaningful characteristics of client local data distributions. In LADD [45], clients apply style extraction to their data through Fourier Domain Adaptation style transfer techniques. Clusters are built by running K-Means over their extracted styles.

5.2. System Heterogeneity

The volatile nature of edge environments combined with the system heterogeneity found across the Computing Continuum makes it challenging to natively support FL. Current approaches rely on client selection schemes to efficiently sample FL clients and other sustainable approaches to better support the execution workflow over heterogeneous devices.

5.2.1. Client Selection Schemes

They can be used to efficiently sample FL clients in order to improve the stability of the FL training or to accelerate the convergence of the model with high quality data.

In [46], authors propose a client selection strategy based on client resource conditions in mobile Edge computing. FedCS defines a 2-steps client selection scheme consisting in asking a set of random clients their resource information (*e.g.*, network state, computational capacities, data size) and selecting them by estimating their required time to achieve a round (including global model download, local training and updated model upload). Authors of FedMarl [48] illustrate the FL problem as a multi-agent reinforcement learning (MARL) problem where the server coordinates a set of RL agents that select in each round the clients participating in the FL training. The server also computes a reward with regard to the test accuracy, total processing latency and communication cost of each round. Authors of [47] address the problem of client selection in volatile environment where the set of available clients changes over time (*i.e.*, clients may join or disconnect) and data collection exhibits discrepancies during the FL training. They propose an online learning scheme that identifies clients with higher utility (effective participation data over the last T rounds). They illustrate their problem as a combinatorial multi-armed bandit and base their solution on the upper confidence bound (UCB) algorithm to select clients with regard to a tradeoff between high utility and low selection rate.

5.2.2. Sustainable Approaches

*Clients in FL are likely to be constrained devices with limited computing resources, network access and battery life. Ensuring the proper support of FL across all devices calls for sustainable approaches (*i.e.*, targeting energy efficiency by minimizing computations and communications).*

Communication-efficiency in FL can be achieved through multiple approaches. Model pruning is used in [49]. Authors propose PruneFL, an algorithm leveraging adaptive model pruning to reduce communication and computation costs resulting in better training efficiency. In [50], two algorithms are proposed. FD (federated distillation) is used to reduce communication overhead through knowledge distillation while FAug uses a generative model to correct the Non-IID nature of local datasets. Their combination results in drastic reduction in communication overhead with little degradation in model accuracy. SPATL [51] uses an RL agent for salient parameter selection of over-parameterized models in order to reduce communication overhead in FL. In SPATL, the model is split into an encoder and a predictor. The encoder part is trained in a federated manner, while the predictor stays local to each client to address data divergence problem among clients. In [52], authors propose FetchSGD where each FL client compresses its model update using Count Sketch, a datastructure that randomly projects a vector several times into lower dimensional spaces such that high magnitude elements can later be recovered. The server then takes advantage of the mergeability of sketches to combine model updates. In [53], authors propose AQFL (Adaptive Quantized FL). After selecting a random subset of clients, the server collects the computational profile of each client. The server then sends custom quantized models to each client with regard to their computational profiles (the model quantization precision is selected to meet the time budget). The server finally aggregates and de-quantizes each local update before updating the global model.

5.2.3. Reducing Latency between Rounds

Network topologies in the continuum and limited computing resources at the edge may lead to high latency and client staleness. Finding the right mechanisms to mitigate these problems avoids slow and inefficient FL training.

A semi-synchronous approach is proposed in [64]. The FL system is split into several FL subprocesses running into clusters of mobile users and orchestrated by small cell base stations. A local FL process is executed in each cluster while all models are periodically synchronized through a macro base station for global consensus. FedAsync [56] minimizes the impact of slow clients by enabling asynchronous updates of the global model and uses an alpha factor to reduce the impact of slow clients (which result in greater error due to staleness). ASO-Fed [57] follows a similar approach by enabling asynchronous updates but also accepts new data arriving during training.

5.3. Robust Systems

Although oriented towards privacy-preserving, FL is not fully secured by nature. Exchanged gradients or model updates may reveal sensitive information to third-parties and might be exploited to reconstruct or infer personal data. As stated in [72], standard FL is vulnerable to several types of attacks (*e.g.*, model poisoning or inference attacks). These vulnerabilities are critical and defensive mechanisms must be adopted to ensure security and user privacy. This problem is exacerbated in the CC where the system is highly and widely distributed. The volatile nature of peers encountered

Table 2
Privacy Preserving Mechanisms for FL

Privacy Preserving Mechanism	Impact on a FL System			Scale
	Accuracy	Computation Overhead	Communication Overhead	
Differential Privacy	Varying (Privacy Trade-off)	No Impact	No Impact	Any
Homomorphic Encryption	No Impact	High	Low	Cross-silo
Secure Multi-party Computation	No Impact	Medium	High	Any
Synthetic data	Varying	Medium	No Impact	Any
Knowledge Distillation	Varying	Medium	No Impact	Any

across the CC adds to this problematic. In such context, the threat level is high. A malicious actor could attempt to take control of a fraction of devices in the federation in order to disrupt the training process (*e.g.*, through adversarial attacks). The system is also subject to privacy-threats through data transmission interception. In Table 2, we listed several mechanisms that can be used to ensure privacy, along with their impact on FL systems and scale of the application, as discussed below.

5.3.1. Privacy Improvement

*One major concern for users privacy in FL is represented by the inference attacks. Since gradients are derived from participants private data, a third-party could infer information about them (*e.g.*, class representatives, membership, properties) or directly infer inputs and labels of the training data (*e.g.*, deep leakage from gradients [73]). We present several mechanisms to limit privacy leakage risks.*

Differential Privacy (DP) is a privacy preserving mechanism used to add random noise to the data. It enables better privacy as data used for training is altered depending on an ϵ factor (the smaller the ϵ factor, the greater the noise on data), yet it reduces training efficiency. Consequently, this factor should be carefully selected to enable privacy while maintaining acceptable model accuracy. Authors from [74] studied the impact of this factor in the FL context and showed that the smaller the ϵ factor (and the greater the noise), the greater the decrease in accuracy for underrepresented classes. This ends in worsening unfairness of the model. Moreover, authors of [75] discuss the limits of DP and highlight its misuse in ML. Notably, using the same data for several epochs will increase the privacy budget and decrease the effective protection. Consequently, using clients data for a smaller number of local epochs is a condition to ensure that DP effectively works.

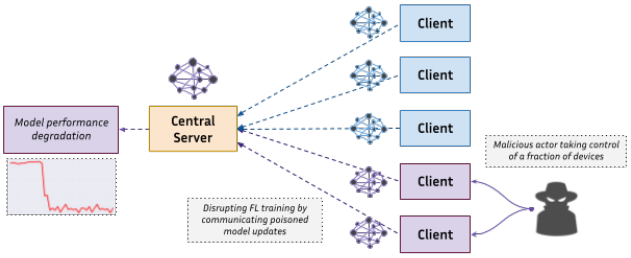
Homomorphic Encryption (HE) enables computations over encrypted data without access to the secret key. In the FL context, HE can be used to let clients encrypt their inputs (locally updated weights) with the HE scheme, afterwhich the server can aggregate encrypted weights and send back the updated model to the clients (which is still encrypted). However, it requires additional expensive computation on the server side for computing over encrypted data. In [76], authors propose a system that combines DP and HE. While HE protects from a semi-honest server by encrypting local model updates, DP protects data from semi-honest clients and end-users that can access the raw model. By introducing a new stochastic quantization operator, they achieve DP guarantees despite the noise being quantized and bounded due to HE.

Secure Multi-Party Computation (SMPC) is a set of cryptographic methods enabling parties to jointly compute a function over private inputs. Although closely related to HE, SMPC relies on multiple communications between parties to provide a computation efficient framework instead of imposing a high computational overhead for the server. However, this requires to design complex communication schemes for synchronization and data exchanges between parties and can lead to high communication overhead. Secret Sharing is a common mechanism used for achieving SMPC. With Secret Sharing, a secret value can be split into n shards such that any t shards can be used to reconstruct the secret value. Any set that contains less than t shards can not give any information about the secret value. In [77], a system based on public key exchanges with signatures and verifications, and secret sharing between clients is used to enable SMPC in the FL setting. By leveraging Secret Sharing, the proposed protocol is robust to user dropouts as long as t users are alive.

Synthetic Data can be used to reproduce the data distributions of participants with limited privacy risks. In [54], a local GAN is trained on each client of a FL system to generate synthetic data based on their data distribution. Laplacian noise is added to the GANs parameters to achieve DP and add additional privacy guarantees. Finally, local GANs are aggregated by the server to generate synthetic data for training.

Knowledge Distillation is the process of using ouputs of an initial model to train a new one. The Private Aggregation of Teacher Ensemble (PATE) [78] approach was proposed for avoiding models to accidentally store training data. N teacher models are trained using N distributed and private datasets. They vote ouputs for nonsensitive and unlabeled data on which a student model is trained. FedKD [55] applies a similar approach in FL. The server aggregates teacher models trained locally by the federated clients to apply knowledge distillation to a student model. To reduce privacy risks, authors propose to use quantization and noise to perturb the teacher outputs.

Figure 6: A poisoning attack in Federated Learning



5.3.2. Securing against Malicious Peers

FL is by nature highly exposed to poisoning attacks. At training time, any client participating in the FL round could introduce malicious updates [79]. Figure 6 illustrates a poisoning attack in Federated Learning scenario. At inferring time, other risks to consider are adversarial examples which are designed by malicious users to intentionally make the model misclassify given inputs. For instance, an attacker could drastically change the predicted output by adding a small noise to a given input. We introduce several approaches to limit these security risks in FL.

Protection against Poisoning Attacks. Several works have proposed to replace the standard FL weighted averaging operator [22] with more statistically robust aggregation operators such as geometric median [80] or norm thresholding [81].

Other works proposed to follow anomaly detection models. FLDetector [61] proposes to detect malicious updates by checking their consistency between rounds. In each new round, the server uses historical data to predict client updates, and flags a client as malicious if its update is inconsistent with the predicted model update. Authors of [58] propose to use conditional variational autoencoders (CVAE) to detect and discard malicious client updates. They pre-train a CVAE from publicly available data and use it to reconstruct client updates. Updates with a high reconstruction error are deemed as malicious and excluded from aggreagtion. In addition they use the geometric median to pre-process client updates in order to defend against same-value attacks. Authors of [62] propose a new protocol for fair detection of poisoning attacks in FL based on micro-aggregations. Their approach consists in collecting demographic attributes from federated peers to build clusters of clients which are expected to produce similar model updates. Following this assumption, peers that provide outlier updates (*i.e.*, updates that are far from the others) in there respective clusters are discarded from the training round. All the remaining updates are used to update the centralized model.

Finally, generative model-based approaches benefit from the generation of synthetic data to evaluate the client's model performance and exclude low performing models from aggregation. In [59], a GAN is trained on the FL server to generate unlabeled synthetic data close to those used for training. The synthesized dataset is then used for evaluation of the client updates. The most frequent predicted labels are selected as the true labels and the client updates that do not meet a minimum threshold accuracy are discarded from the current iteration. Authors of [60] propose to synthesize validation data from CVAEs trained locally by the federated peers. This approach benefits from the ability of CVAEs to condition their outputs on specific labels to produce balanced and labeled validation dataset.

Adversarial Examples. In [82], authors leverage DP properties to enable model robustness against adversarial examples. They reduce the sensitivity of the DNN model by breaking it down in two parts and add a noise layer between the two.

In [63], authors propose to improve the adversarial robustness of models in the FL context by introducing adversarial examples during the training phase. For this, the server uses bias-variance based adversarial attacks to build adversarial examples with large bias and variance with regard to clients local models. These adversarial examples are sent to the clients and used for local training in addition to their local data to produce robust local models.

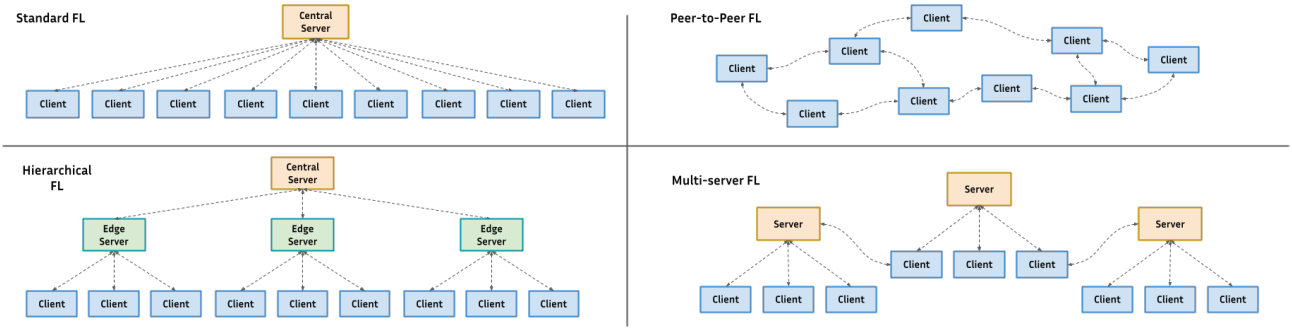
5.4. Interconnecting Peers across the CC

The communication channels between the server and clients in the Edge may suffer from quality degradation due to distant communications or interference. In this context, several approaches exploring different topologies for FL systems were proposed to improve communication-efficiency and privacy (Figure 7). While the introduction of relay gateways can reduce communication-cost for Edge clients, exploring alternative topologies can further improve privacy.

5.4.1. Hierarchical FL

Hierarchical FL (HFL) introduces intermediate layers (at least one) between the Cloud server and the Edge clients. Devices in these intermediate layers are used as relay gateways to increase the network coverage area through multi-hop routing. Instead of performing long and expensive communications with the central server, Edge clients communicate their model updates to proxy gateways which take care of forwarding them to the central server [64]. HierFAVG [65] introduces a semi-synchronous HFL approach. Each Edge server performs several rounds of partial aggregation and synchronizes periodically with the central server for global aggregation of the model. Authors show that increasing the Edge aggregation frequency can reduce energy consumption by improving model convergence. On the other hand, too frequent synchronization may also result in the opposite effect. Authors of [66] propose a multi-level (*i.e.*, more than 3 layers) HFL framework, significantly increasing the scalability of the system to wide geographic areas. They propose a greedy algorithm based on model aggregation and training time estimation to perform a stratified participant selection. A topology learning approach for HFL has been introduced by [67]. Authors propose to optimize: (1) communication-efficiency by minimizing the distance between clients and edge aggregators, and (2) model convergence by shaping balanced data distribution under a same edge aggregator.

Figure 7: FL Topologies



5.4.2. *Peer-to-peer FL*

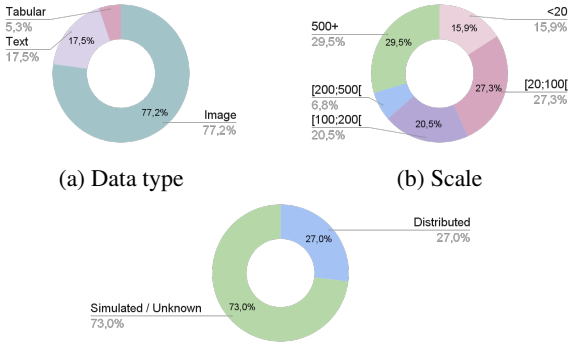
Fully Decentralized Approaches can be of interest to improve privacy by avoiding communication of local updates to a central entity. Instead, federated peers use peer-to-peer protocols to jointly train the model. In [68], each participant owns a private model that stays local and a public (or proxy) model that is shared with neighbor participants. All participants agree on a proxy model architecture which usually is smaller than the private models. Proxy models enable efficient information exchange between clients with DP guarantees which improves privacy and reduces communication overhead. Authors of [69] explore PFL in decentralized setting. They propose Dis-PFL, a peer-to-peer approach using sparse masks to train and communicate local models with little communication and computation overhead. One challenge coming from such fully decentralized approaches is that the training efficiency is highly correlated with the topology of the network. Some recent works have been targeting this problem in decentralized FL. For instance, [83] proposes to learn a network topology that optimizes the neighborhood of each client with respect to data heterogeneity to efficiently train the model.

5.4.3. *Multi-server FL*

Multi-server FL has been studied to improve model performance with personalization through clustered FL [41, 42, 43, 44, 45] (discussed in section 5.1.2). Other works have considered multi-server FL with a spatiality constraint, deploying seperated FL training based on the coverage areas of different edge servers. [70] specifically focus on multi-server FL with overlapping areas between edge servers (*i.e.*, clients could be in the intersection of multiple servers). Clients in overlapping areas enable knowledge sharing across the network by spreading the models of the different edge servers across the multiple subnetworks.

5.5. Discussion

Overall, the statistical heterogeneity, system heterogeneity and privacy problems have been active research topics in FL, resulting in significant progress in these directions. **Personalized FL** is a promising direction to address statistical heterogeneity. Compression schemes such as model **quantization** and model **pruning** enable a broader support of



(a) Data type (b) Scale

(c) Experimental platform

Figure 8: Evaluation settings used in the reviewed studies

devices with limited resources, and **client selection schemes** are proposed to better sample quality participants in volatile environments. Regarding privacy, one already well adopted mechanism is **differential privacy**, while other approaches such as **SMPC and HE** are current research directions.

Recent efforts have been made towards more decentralized approaches to deploy FL. Hierarchical, peer-to-peer and multi-server FL have been explored to improve model convergence (*e.g.*, through clustered FL), communication-efficiency (*e.g.*, using relay gateways) and privacy (*e.g.*, by eliminating the central orchestrator).

Finally, we highlight that the common practice in FL is to perform evaluation through simulation: 73% of the reviewed studies ran experimental evaluation using simulation platforms (Figure 8c). Concerning the use-cases, FL studies have mainly considered computer vision tasks (Figure 8a) - 77.2% of the evaluation tasks, while natural language processing represents 17.5% and tabular data processing represents 5.3% of the evaluation tasks. We observe a certain diversity regarding the scale of experiments (Figure 8b). While some studies consider very small scale FL deployments (<20 clients), others scale to more than 500 clients.

Table 3
Major state-of-the-art FL Frameworks and Benchmarks

	FL Setting		Topology			Training mode		Can be interfaced with		
Framework	Horizontal	Vertical	Standard	Hierarchical	P2P	Simulation	Distributed	TensorFlow	PyTorch	Other Framework
TFF [84]	✓	✗	✓	✗	✗	✓	✗	✓	✗	-
Flower [85]	✓	✗	✓	✗	✗	✓	✓	✓	✓	Agnostic
OpenFL [86]	✓	✗	✓	✗	✗	✗	✓	✓	✓	-
FATE [87]	✓	✓	✓	✗	✗	✓	✓	✓	✓	EggRoll, Spark
Substra [88]	✓	✗	✓	✗	✗	✓	✓	✗	✓	-
IBM FL [89]	✓	✗	✓	✗	✗	✓	✓	✓	✓	Agnostic
PaddleFL [90]	✓	✓	✓	✗	✗	✓	✓	✗	✗	PaddlePaddle
Nvidia FLARE [91]	✓	✗	✓	✗	✗	✓	✓	✓	✓	Agnostic
FedTree [92]	✓	✓	✓	✗	✗	✓	✓	✗	✗	ThunderGBM
FLSim [93]	✓	✗	✓	✗	✗	✓	✗	✗	✓	-
FLAME [94]	✓	✗	✓	✓	✗	✓	✓	✓	✓	-
p2pfl [95]	✓	✗	✗	✗	✓	✓	✓	✗	✓	-
Benchmark	Horizontal	Vertical	Standard	Hierarchical	P2P	Simulation	Distributed	TensorFlow	PyTorch	Other Framework
FedJAX [96]	✓	✗	✓	✗	✗	✓	✗	✗	✗	JAX
FedTorch [97]	✓	✗	✓	✗	✗	✓	✗	✗	✓	-
FedScale [98]	✓	✗	✓	✗	✗	✓	✓	✓	✓	-
LEAF [99]	✓	✗	✓	✗	✗	✓	✗	✓	✗	-
FedNLP [100]	✓	✗	✓	✗	✗	✓	✓	✓	✓	-
OARF [101]	✓	✓	✓	✗	✗	✓	✗	✗	✓	-
NIID-Bench [102]	✓	✗	✓	✗	✗	✓	✗	✗	✓	-
FLamby [103]	✓	✗	✓	✗	✗	✓	✗	✗	✓	-
UniFed [104]	✓	✓	✓	✗	✗	✓	✗	✓	✓	Agnostic

6. Taxonomy of FL Frameworks, Benchmarks and Performance Optimisation Tools

In this section we present a structured view of the existing frameworks and benchmarks according to the approaches discussed in the previous section. We further provide details about the right choice of FL frameworks, their configuration and monitoring for an efficient execution on the Computing Continuum.

In Table 3 we list the major frameworks and benchmarks for FL and give details about their supported FL settings, topology, training modes and ML frameworks they can interface with or they depend on. *Training mode* refers to whether the framework is simulating the federated process using one or few nodes, or it provides a mechanism to distribute the FL process on many nodes.

6.1. Frameworks

Several frameworks have been proposed to help users in developing their FL systems. While simulation frameworks enable fast FL simulation for prototyping on a single machine, distributed frameworks enable a more realistic experience through the deployment of the FL server and clients on distributed nodes.

Simulation frameworks. Tensorflow Federated [84] from Google provides a single simulation environments composed of two main layers. Federated Learning API is a high-level interface used by developers to apply FL algorithms to their TensorFlow models and data while Federated Core API provides a low-level interface enabling developers to design new algorithms.

FLSim [93] is provided by Meta Research as an easy-to-use FL Simulator using PyTorch. By default, FLSim core is using FedAvg, however the server and local optimizers can easily be changed by configuring values of a JSON object (*e.g.*, changing SGD to FedProx or synchronous to asynchronous aggregation). Likewise, the number of global and local epochs, or the number of sampled clients per round can be modified. Moreover, FLSim supports several privacy mechanisms including differential privacy and secure aggregation, and lets the researcher define a metric reporter to collect relevant metrics and log them into TensorBoard.

Distributed frameworks. Flower [85] is a framework for cross-device horizontal FL. It can be used with any ML framework and comes with a well-furnished documentation. Flower enables researchers to design new algorithms by describing global logic (*e.g.*, client selection, aggregation, federated or centralized evaluation) and local logic (*e.g.*, local model training and evaluation). Researchers may also choose between realistic multi-node FL or single-machine simulation with Ray to run their experiments.

OpenFL [86] provides a Python API and a CLI to perform FL and let the developer choose between Director-Based Workflow which uses long-lived components to run many experiments in series easing FL research or Aggregator-Based Workflow to run a unique federation with short-lived components. To ensure security of the framework, OpenFL uses Public Key Infrastructure Certificates and provides

a Trusted Execution Environments to execute code with various security properties.

FATE [87] targets Industrial FL. It aims at providing all the necessary tools for production environments. FederatedML lets developers access many implementations of algorithms for vertical and horizontal FL as well as federated statistics (*e.g.*, Private Set Intersection), federated feature engineering (*e.g.*, Feature Sampling) and secure protocols. Moreover, FATE comes with several tools to help developers in their tasks: FATE-Flow coordinates the execution of algorithmic components, FATE-Board is a visualization tool used to monitor the federated system, FATE-Serving is used to improve federated inference, FATE-Cloud enables federated site and client management, and KubeFATE enables the use of FATE with Kubernetes.

Substra [88] is a FL framework for clinical data. It provides 3 types of interfaces to interact with: a web interface to trace all operations and modifying permissions of the different organizations, a CLI to add new datasets, algorithms or objectives, and a Python SDK to let developers integrate Substra in their applications. Substra uses a distributed ledger to trace all operations and uses kubernetes to run the workers and server on distributed nodes. Substra proposes 3 training modes: local mode for simulation on a single node, deployed mode for running all the tasks on a deployed Substra platform and hybrid mode where tasks run locally using assets from remote organizations.

IBM FL [89] is a framework for Industrial FL which can be used with any ML framework. It supports fusion algorithms for neural networks, decision trees, reinforcement learning, linear classifiers, K-means and Naïve Bayes. An Experiment Manager Dashboard is provided to configure, run and monitor FL experiments while IBMFL Multi-Cloud Orchestrator is used to automate the deployment and monitoring on OpenShift clusters.

FedTree [92] is a FL framework for tree-based models in vertical and horizontal settings providing CLI and Python interface. It allows configuration of privacy preserving mechanisms such as Homomorphic Encryption (for vertical setting), Secure aggregation (for horizontal setting) and Differential Privacy. FedTree allows standalone simulation or distributed training mode.

PaddleFL [90] is a framework based on PaddlePaddle from Baidu. PaddleFL provides two components to perform FL: Data Parallel component enables distributed nodes to train a model using common horizontal FL strategies while PFM component which supports horizontal, vertical and transfer learning scenarios uses secure multi-party computation to secure training and predictions.

Nvidia FLARE [91] provides a framework-agnostic Python SDK for FL. It comes with privacy preserving mechanisms (*e.g.*, Differential Privacy, Homomorphic Encryption). A simulator can be used for prototyping on a single node and FLARE Console lets administrators manage the system (*e.g.*, submitting jobs, starting or stopping clients) with a CLI or a FLARE Dashboard can be used for simplified project management.

Cisco proposes FLAME [94] to provide fine grain abstraction for the composability and extensibility of FL tasks. With FLAME, FL applications are described as topology abstraction graphs (TAGs) to decouple the ML application logic from the deployment details. FLAME lets developers define roles and channels. The role abstraction defines a set of functions for training and evaluating the model, loading data, getting/distributing model updates from/to another role. The channel abstraction, on its part, creates links between roles to enable the exchange of data between them. Moreover, FLAME provides a channel manager interface coming with a set of APIs accessible by any role ensuring the compatibility with many communication backends (*e.g.*, MPI, MQTT, Kafka, gRPC).

p2pfl [95] is a framework for decentralized FL systems. It enables the construction of peer-to-peer networks using gossip protocols with gRPC and FL training using FedAvg.

6.2. Benchmarks

FedJAX [96] is provided as a framework for fast simulation of FL experiments. It uses JAX to take advantage of accelerators such as GPUs and TPUs. FedJAX comes packaged with CIFAR-100, EMNIST, Shakespeare and Stack Overflow datasets with baseline models while also providing tools to create new datasets and models. Implementation of FedAvg and clustering algorithms are provided with the framework.

FedTorch [97] is a Python FL library built on top of PyTorch distributed API. It comes with implementation of several optimization algorithms including FedAvg, SCAFFOLD, FedProx and APFL among others. It supports EMNIST, Shakespeare, Adult and Synthetic datasets and proposes a mechanism to distribute a dataset and make it Non-IID.

FedScale [98] is a benchmark focusing on cross-device mobile FL which can be used with PyTorch and Tensorflow providing high level APIs to implement FL algorithms. It comes with more than 20 large-scale Non-IID federated datasets consisting in computer vision, natural language processing and other applications. FedScale proposes two backends: a mobile backend for on-device FL evaluation and a cluster backend for training thousands of clients using few GPUs. Moreover FedScale includes environment datasets to emulate heterogeneous system performance and availability of clients.

LEAF [99] is a benchmark for cross-device FL providing a suite of open-source datasets simulating federated scenarios: *FEMNIST, Sentiment140, Shakespeare, Reddit and a Synthetic dataset* and implementations of *Minibatch SGD, FedAvg and Mocha*. LEAF enables the user to monitor statistical metrics (performance at a given time) and system metrics (computing resources needed from edge devices and number of bytes downloaded and updated).

FedNLP [100] is a benchmark for natural language processing tasks. It is part of the FedML ecosystem and uses it to provide implementations of FedAvg, FedProx and FedOpt. FedNLP divides NLP applications into 4 categories and

provides a dataset for each of them: 20Newsgroup for Text Classification, OntoNotes for Sequence Tagging, MRQA for Question Answering and Gigaword for Seq2Seq Generation. Each dataset is horizontally partitioned following Non-IID schemes.

OARF Benchmark Suite [101] enables benchmarking of horizontal and vertical FL systems by providing datasets for both settings. For horizontal FL, several datasets for Computer Vision, NLP and Geographic Information System (GIS) tasks are provided. For vertical FL, provided datasets represent recommendation and year prediction tasks. These datasets are partitioned using Dirichlet distribution to simulate label distribution skew and quantity skew. OARF also provides implementation of FL algorithms (*e.g.*, FedAvg, FedProx) and allows comparison of algorithms using performance (*e.g.*, accuracy, time to convergence) and system metrics (*e.g.*, communication cost).

NIID-Bench [102] enables benchmarking of FL algorithms under Non-IID data distribution scenarios. It provides implementation of FedAvg, FedProx, SCAFFOLD and FedNova and lets researchers compare algorithms using 9 datasets with several Non-IID settings: Label distribution skew, Feature distribution skew or Quantity skew.

FLamby [103] is a recent benchmark focusing on cross-silo FL in realistic healthcare settings. It is provided with 7 datasets covering classification, segmentation and survival tasks with natural splits and baseline models. Interfacing with other FL frameworks should be easy as examples of Fed-BioMed, FedML and Substra interface are provided with the benchmark. Common FL strategies are provided such as FedAvg, FedProx, SCAFFOLD and FedOpt.

UniFed [104] is focusing on the selection of a FL framework. It enables evaluation of FL frameworks with scenarios for cross-device and cross-silo horizontal FL as well as vertical FL. To this end, performance of a framework is defined based on its training efficiency, communication cost and resource consumption.

Selecting the right framework. While numerous tools for implementing FL algorithms are provided, selecting the best tool is not easy and many factors need to be taken into account. For instance, by looking at Table 3, one can quickly identify a subset of frameworks that best fit some specific needs by filtering the FL setting, training mode and framework interface. However, while identifying corresponding frameworks, this do not tell which is performing the best.

Authors from [105] proposed a comparative review of several FL frameworks including TFF, FATE and Paddle FL. In this review, authors compared accuracy and training time of these frameworks and could draw first conclusions on their performance. However system metrics such as energy consumption or bandwidth usage are very important to monitor in the FL context considering that FL clients could be resource-constrained devices. As presented earlier, UniFed [104] is a benchmarking tool for FL frameworks that lets researchers compare frameworks performance (training efficiency, communication cost and resource consumption) on several scenarios. Such tool could be used to compare training and system performance of many frameworks, therefore helping in their selection. Nevertheless, ease of use or deployment aspects are also important features that should be considered when selecting a framework. Flexibility is another important factor to consider. While the reviewed frameworks mainly support the standard client-server parameter server topology, FLAME [94] is the only framework that enables a flexible configuration of the roles and communications of the FL system (an important feature to develop decentralized approaches).

6.3. Hyperparameter Tuning

Hyperparameter optimization (HPO) can be performed either by simulation or in a distributed manner during the FL process. Simulations benefit from HPC accelerators to efficiently explore many FL configurations and to get fast results. Distributed HPO enables a more realistic evaluation of the different configurations by monitoring the computing resources and the communication. HPO may target several objectives (*e.g.*, accuracy of the model, resource consumption) by tuning a broad range of parameters (*e.g.*, model architecture, number of local epochs, client amounts selected for each round, optimization algorithms).

In [106], authors advocate that commonly used models in centralized settings may not be the optimal choice for Non-IID settings encountered in FL. They propose to perform Neural Architecture Search in the FL setting where each client performs a local searching process.

Authors of [107] introduce an approach for single-shot FL HPO. Each client starts by performing local HPO asynchronously using adaptive HPO schemes (*e.g.*, Bayesian Optimization) to efficiently approximate loss surfaces of the hyperparameter space with low communication overhead. Then, the server aggregates the local loss surfaces and builds a model to predict the best hyperparameter configuration to fit the population. Finally, a FL process is started to train a shared model using the best hyperparameter candidate.

In [108], authors proposed a system to automatically tune FL hyperparameters during the training process given optimization preferences. Authors define system metrics (*e.g.,* computation time, transmission time, computation load and transmission load) to optimize and let the users select their optimization preference by giving priorities to each of the metrics. The system then updates hyperparameters (*e.g.*, number of participants, number of training passes) given user preferences by targeting higher weighted improvements over the weighted degradation for each optimization step.

In a preliminary work, authors of [109] propose a system design for optimizing FL systems in the CC. They propose to use a formal description of the experimental infrastructure (large-scale testbeds) to deploy several FL configurations in parallel among a population of heterogeneous devices and perform distributed HPO using an exploration strategy (*e.g.*, Grid Search, Random Search, Bayesian Optimization).

6.4. Monitoring Model Performance

To ensure that the trained model is performing as wanted, the model performance needs to be evaluated using several metrics collected during training (*i.e.*, training and system metrics). In the FL context, the model can be evaluated either on the server side, using a fix test set, or locally using clients data in a federated manner.

The impact of personalized FL is evaluated in [110] through on-device evaluation. Federated clients receive a copy of the global model and apply local fine-tuning to adapt the model to their local data distributions. They use their local test data to evaluate the global model and the locally fine-tuned model, and measure the performance improvement.

Authors of [111] report that most personalized FL studies only evaluate their models with training loss, average validation accuracy and prediction error metrics. However, reporting the *per-user* performance is also important to identify potential bias in such scenarios. They propose a set of metrics for assessing performance of personalized FL methods *e.g.*, *Percentage of User-models Improved* and fairness metrics *e.g.*, *Average variance*.

While traditional metrics such as model accuracy and resource consumption are generally considered in FL, more specific metrics may also be useful. For instance, measuring user contributions can be an important factor to better select high quality participants and reward them accordingly. In [112], authors leverage such metrics to detect undesirable user strategies such as an overly privacy-preserving technique adding excessive noise to the model which actually impairs the training phase.

Provenance capture can benefit FL systems by helping in their optimization, interpretability and explainability. This is particularly important and challenging in the CC (with heterogeneous and volatile participants). Capturing which hyperparameter combinations were used by the federated peers during training can assist in the interpretability of model performance. Specifically, a model performance degradation could be the cause of a wrong hyperparameter selection or the sign of a data corruption at a specific client. Some works specifically focus on this aspect. In [113], task results and telemetry metrics are aggregated with noised model updates by the server. High level statistics such as average task execution times and amount of on-device data are aggregated by the server while sensitive metadata coming from the HTTP communication with the users are dropped (*e.g.*, IP addresses). ProvFL [114] introduces a neuron provenance mechanism in FL. It dynamically isolates sensitive neurons of the model and extracts each client's contribution to the selected neurons, enabling accurate identification of clients responsible for the given behaviour of the global model.

In most of these studies authors consider testing the model directly with clients data and aggregating FL metrics to the server. However, sharing raw training metrics with the server can induce a privacy risk. Knowing that the model performed well or poorly on a given client can give hints about training data of that specific client to an attacker. Therefore, there is a need to think how to best aggregate these metrics with respect to clients privacy, or to consider alternatives such as model testing on the server side (*e.g.*, through synthetic data). System metrics on the other hand, do not give information about the training inputs, therefore can be directly aggregated to the server.

Monitoring tools. Traditional monitoring tools such as Tensorboard [115], Neptune [116], Weights & Biases [117] or MLflow Tracking API [118] provide visualization components for training metrics of ML experiments (*e.g.,* loss, accuracy, model graph, model weights). More specialized tools such as FATE-Board [87] or FedML MLOps platform [119] have been proposed to better capture FL metrics. FATE-Board offers data and model visualization tools for tracking the progress of the FL jobs, dataset information and model outputs. The FedML MLOps platform provides tools to visualize the FL topology, device status, training results (*e.g.*, accuracy and loss) and system performance (*e.g.*, CPU and memory utilization). More specialized tools for system performance monitoring are available. For instance, Telegraf [120], InfluxDB [121] and Grafana [122] can be combined together to retrieve and visualize system metrics of distributed clients. Telegraf agents collect metrics from each client (*e.g,* GPU, CPU, memory and bandwidth usage) and store them into time series databases using InfluxDB. Grafana enables visualization of these metrics through dynamic dashboards accessible from a web interface. ProvLight [123] is proposed for efficient provenance capture on the IoT/Edge. Data compression and grouping coupled with the use of lightweight communication protocols enable low provenance capture overhead.

6.5. Discussion

Defining impacting metrics that capture well the behaviour of the system is essential to get a global understanding of its performance (*e.g.*, resource utilization, per-user accuracy, variance). Coupling those metrics with the possibility to select optimization preferences, as proposed in [108], is a promising direction for multi-objective optimization in FL. Unfortunately, although **aggregating per-user metrics** enables a better understanding of the performance of the system across the different clients, it can also induce a **privacy risk** (*e.g.*, performance of the model on a specific client can give hints about its local data). Therefore, one should also investigate **how to best aggregate**, or choose those metrics to minimize privacy risks. Applying **differential privacy to aggregated metrics** could be a solution to a certain extent. In addition, FL studies usually consider model performance and resource utilization as separate metrics. Most of the previous studies have monitored model convergence based on the number of rounds - an approach which may introduce biases, as different strategies could result in very different resource usage per training round. A better practice for FL experiments is to study model convergence based on resource usage (*e.g.*, total communication to achieve a target accuracy).

7. Deployment and Supervision of FL Systems across the Continuum

In this section we present several tools to facilitate model management and deployment of FL systems. These automatic deployment tools are particularly useful for **reproducibility** purposes - a long-desired goal of the AI, HPC and Big Data communities. This means allowing researchers to accurately reproduce relevant behaviors and representative settings of a given FL application in a controlled environment, through extensive experiments in a large-enough spectrum of potential configurations of the underlying Edge-Fog-Cloud infrastructure. This turns to be a non-trivial endeavour due to the multiple combination possibilities of heterogeneous hardware and software resources, system components for data processing, data analytics or FL model training.

7.1. Model Management

Several tools have been proposed to facilitate the management of FL models through centralized storage, model versioning and provenance capture. In this section, we take a look at some of these approaches.

MLflow Model Registry [118] is a centralized model store that comes with many features for model life cycle management. For instance, by using MLflow Model Registry, developers can assure model versioning, model lineage (retrieving the inputs used for the experiment that produced a given model *e.g.*, hyperparameters, algorithm, libraries), stage transition (transitioning from staging to production or archiving a model) and model serving. Weights & Biases Model Registry [117] proposes another option for model management. It lets developers do model versioning, model lineage, highlight the best model versions evaluated for production and accessing a history of all changes.

Besides providing a well furnished cloud service with ML training features, Amazon SageMaker [124] comes with an Edge Manager component that can be used for compilation, packaging and deployment of the model on edge devices. A Model Registry component also allows to store trained models, make model versioning and automate deployments. Some FL frameworks also provides such features. For instance, FATE [87] and FedML [119] support model serving through FATE-Serving and FedML MLOps components.

7.2. Deployment Tools

Deployments on the Edge-Fog-Cloud Continuum require flexible tools to configure and manage heterogeneous workflows over distributed resources.

KubeFlow [125] is a toolkit that enables deployment of containerized ML applications using Kubernetes. With KubeFlow, one can easily customize the platform and services used in each stage of the ML workflow (*e.g.*, data preparation, model training). KubeFATE, which is based on KubeFlow, is the component of FATE framework [87] for deploying and managing FATE clusters. FATE also supports automatic deployments using Ansible or manual deployment with a CLI. With FedML MLOps component [119], one can build deployable packages for the FL server and clients, and easily deploy the application by accessing a web interface. IBM FL [89] uses a multi-cloud orchestrator to automatically deploy and monitor FL experiments on OpenShift clusters. E2Clab [126] is a deployment tool for reproducible experiments on large-scale testbeds (*e.g.*, Grid5000 [127], FIT IoT-LAB [128], Chameleon [129]) with a focus on the Computing Continuum [130] (edge-fog-cloud environment). It enables researchers to easily deploy, monitor, and reproduce distributed experiments using description files of the experiment workflows and resources.

7.3. Discussion

ML Operations (MLOps) is a set of practices, usually adopted by companies, targeting efficient model deployment and model maintenance in production. MLOps is now a well-established field with many tools that have already been provided. However, new needs are emerging towards highly distributed and heterogeneous infrastructures, especially in the context of **deployments across the Computing Continuum**. Furthermore, with new data regulations, privacy preservation is now emerging as a major concern, which FL tries to achieve. New practices and tools should be further developed to enable efficient MLOps in the FL context (*i.e.*, **Federated MLOps**) as decentralized storage and processing add new constraints for the training of ML models [94]. Along these lines, support for better composability and diverse topologies for FL tasks will drive its broader adoption across the Continuum.

8. Open Issues and Research Opportunities

Federated Learning is a fairly new paradigm for distributed ML, which has been quickly adopted by the community and it is currently a very active research topic. As shown in this survey, many research efforts focus on improving privacy and performance of FL in heterogeneous settings. We identified several other research directions that have not yet been explored or have received little attention in the field of FL, in order to facilitate its adoption across the continuum. In this section, we discuss some of them.

8.1. Performance Evaluation in Realistic Environments

Most of the current research efforts are using simulation environments to evaluate performance of their FL systems (*e.g.*, using one or few nodes with GPUs). While enabling fast results through HPC accelerators, this approach does not allow for real capture of system performance (*e.g.*, bandwidth usage, power consumption) and other factors that impact on the model in real-life settings. Using real devices in distributed settings is needed to get realistic evaluations of FL system performance. Scientific testbeds such as Grid5000 [127] and Chameleon [129] offer more and more supports for such experiments by providing a large collection of reconfigurable devices. Public clouds such as Amazon Web Services or Microsoft Azure are other alternatives that supports distributed experiments. Reconfigurable platforms

enable the simulation of real network topology such as the ones shared by [131]. It is important to simulate such networks to better understand the applicability of different FL approaches in constrained and realistic environments.

8.2. Reconfigurable Systems

Enabling dynamic deployment and configuration of the system based on new objectives and environment changes will result in better FL support for complex environments. For instance, such a reconfiguration could be needed in case of clients deprived of network accessibility. In this situation, by dynamically reconfiguring the system, clients could perform additional local training epochs until network recovery. Topology learning approaches can help overcome these problems. Building clusters based on similarity in data distributions to improve model convergence on federated clients (*e.g.*, cluster personalization), or dynamically deploying relay gateways close to the edge to improve the communication-efficiency (*e.g.*, hierarchical FL) are some example of such initiatives.

8.3. Addressing the Concept Drift

Data drift between clients has been well studied in the FL setting. On the other hand, concept drift, which translates into sudden changes in the relationship between the inputs and expected outputs of the model, is a direction that has been little explored, despite its important impact on model performance.

Key research challenges in this context are how to detect concept drift and how to react to it. *Clustered FL* address this problem to a certain extent by grouping clients with similar data distributions. Still, current approaches lack migration mechanism to dynamically adapt in time-varying clustering problems (*e.g.*, moving clients from one cluster to another when their local data distributions are subject to conceptual drift during training). New approaches are required to detect drifted clients and identify candidate clusters for migration. First efforts have been made in this direction in [132]. *Continual learning* proposes an alternative solution by incrementally training the model using the continuous flow of data. An important challenge in continual learning is the problem of *catastrophic forgetting* which requires innovative mechanism to ensure that previously learned patterns are not forgotten. FL could benefit from continual learning to address the concept drift problem. However high variability in terms of clients and training data across the CC makes it an important challenge.

8.4. Enabling Seamless Composability

Many components are put together in order to build a complete FL system (*e.g.*, algorithms, privacy, security). Combining these building blocks into an efficient application is non-trivial. Yet, there is a need for such highly composable FL systems as the various component combinations satisfy specific and changing requirements, as shown in this survey.

Algorithms, privacy and security mechanisms should be designed in a way that makes them easy to implement and interface with each other. Similarly, FL frameworks should be easy to interface with other tools to avoid unnecessary extra configuration of the application.

Existing FL frameworks mainly consider the standard Edge-Cloud parameter-server architecture and do not provide flexible configuration to deploy alternative FL network topologies (*e.g.*, hierarchical or peer-to-peer). Deployments on the CC require flexible tools to describe heterogeneous workflows which is currently missing from most FL frameworks.

A first step in preparing for composability is to enable its core attributes, identified by Gartner [133]: modularity (components should have a singular, clear and complete functionality and be able to operate as standalone services); discovery (easily identifiable and accessible components); autonomy (easily changeable and self-contained building blocks); and orchestration support (enabling easy interaction with other applications and services). These attributes can be enabled by familiar technology, from APIs to containers.

8.5. Explainability Support

In critical systems that rely on AI to take decisions, it is crucial to understand why a model is taking a decision in a specific context. Using models that are easily interpretable such as linear regressions or decision trees can be a solution. However, opaque models such as random forests or deep neural networks which are more challenging to interpret might be needed to capture the complex patterns of a targeted task.

Current techniques for interpreting those models rely on explanation by simplification, feature relevance and visualization [134]. Investigating how those techniques can be adapted in the highly distributed FL setting will enable the use of FL in critical applications. Provenance capture is also helping by tracking hyperparameter combinations used by the federated peers. This leads to a better interpretation of the model performance, drives the optimization and supports reproducibility of experiments across the CC.

8.6. Sustainability

Current trends in ML are moving towards two opposite directions. On the one hand, energy consumption is becoming a major topic in daily life with the rising cost of energy as well as its environmental impact. On the other hand we are continuously executing more complex tasks (*i.e.*, training more complex models). In both cases, building energy efficient systems (*e.g*, through model compression) will enable a better support of resource-constrained devices as well as training more complex models in FL. More informed model selection coupled with model pruning [49] or model quantization [135] are some directions to achieve better sustainability.

8.7. System Robustness vs. Fairness

One problem arising from the decentralized and heterogeneous nature of FL on the continuum is that it is vulnerable to various threats coming from federated peers such as poisoning attacks. Current approaches for detection of malicious peers may filter out benign minority groups

that have very different data compared to the rest of the participating peers. While minority groups allow for better data representativity in FL, they are usually identified as outliers and are discarded from the training rounds. As a result, one important challenge is to effectively distinguish minority groups from malicious peers to enable robust and bias-free FL [136]. Geographic and demographic data [62] could support such initiative by performing stratified detections of malicious peers.

9. Conclusion

In this survey we make an in-depth analysis of the challenges related to the support of FL across the Computing Continuum. We review state-of-the-art FL systems addressing system heterogeneity and statistical heterogeneity as well as enabling better privacy. We analyze existing tools for implementing, monitoring, configuring and deploying such systems. Finally, we discuss several major open issues and research opportunities to better support FL on the Edge-to-Cloud Continuum.

Investigating how to best address these challenges will lead FL to a mature stage and help in its adoption by a broader community. This will enable the use of FL in complex systems that are restricted by various regulations. Smart cities, defense equipment and smart medical devices are some of the numerous examples of systems that could benefit from such future advances.

CRediT authorship contribution statement

Cédric Prigent: Conceptualization, Investigation, Visualization, Writing - Original Draft. **Alexandru Costan:** Writing - Review & Editing . **Gabriel Antoniu:** Funding acquisition, Resources, Writing - Review & Editing . **Loïc Cudennec:** Writing - Review & Editing .

Acknowledgement

This work was funded by the ENGAGE Inria-DFKI project. In addition, as part of the "France 2030" initiative, this work has benefited from a State grant managed by the French National Research Agency (Agence Nationale de la Recherche) attributed to the STEEL project of the CLOUD PEPR program, reference : ANR-23-PECL-0007.

References

[1] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y. Zomaya. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469, 2020.

[2] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, et al. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019.

[3] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *CoRR*, abs/1902.04885, 2019.

[4] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[5] Qiang Duan, Jun Huang, Shijing Hu, Ruijun Deng, Zhihui Lu, and Shui Yu. Combining federated learning and edge computing toward ubiquitous intelligence in 6g network: Challenges, recent advances, and future directions. *IEEE Communications Surveys and Tutorials*, 25(4):2892–2950, 2023. Publisher Copyright: © ; 2023 IEEE.

[6] Haftay Gebreslasie Abreha, Mohammad Hayajneh, and Mohamed Adel Serhani. Federated learning in edge computing: A systematic survey. *Sensors*, 22(2), 2022.

[7] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *CoRR*, abs/1909.11875, 2019.

[8] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey. *CoRR*, abs/2106.06843, 2021.

[9] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Sha Wei, Fan Wu, Guihai Chen, and Thilina Ranbaduge. Vertical federated learning: Challenges, methodologies and experiments, 2022.

[10] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M. Hadi Amini. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24, 2022.

[11] Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. From distributed machine learning to federated learning: A survey. *CoRR*, abs/2104.14362, 2021.

[12] Sin Kit Lo, Qinghua Lu, Liming Zhu, Hye-young Paik, Xiwei Xu, and Chen Wang. Architectural patterns for the design of federated learning systems. *CoRR*, abs/2101.02373, 2021.

[13] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *CoRR*, abs/1907.09693, 2019.

[14] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *CoRR*, abs/2003.02133, 2020.

[15] Dinh C. Nguyen, Quoc-Viet Pham, Pubudu N. Pathirana, Ming Ding, Aruna Seneviratne, Zihuai Lin, Octavia A. Dobre, and Won-Joo Hwang. Federated learning for smart healthcare: A survey. *CoRR*, abs/2111.08834, 2021.

[16] Mansoor Ali, Faisal Naeem, Muhammad Adnan Tariq, and Geroges Kaddoum. Federated learning for privacy preservation in smart healthcare systems: A comprehensive survey. *ArXiv*, abs/2203.09702, 2022.

[17] Dashan Gao, Xin Yao, and Qiang Yang. A survey on heterogeneous federated learning, 2022.

[18] Edoardo Gabrielli, Giovanni Pica, and Gabriele Tolomei. A survey on decentralized federated learning, 2023.

[19] Audris Arzovs, Janis Judvaitis, Krisjanis Nesenbergs, and Leo Selavo. Distributed learning in the iot–edge–cloud continuum. *Machine Learning and Knowledge Extraction*, 6(1):283–315, 2024.

[20] Daniel Rosendo, Alexandru Costan, Patrick Valduriez, and Gabriel Antoniu. Distributed intelligence on the edge-to-cloud continuum: A systematic literature review. *Journal of Parallel and Distributed Computing*, 166:71–94, 2022.

[21] Luiz Bittencourt, Roger Immich, Rizos Sakellariou, Nelson Fonseca, Edmundo Madeira, Marilia Curado, Leandro Villas, Luiz DaSilva, Craig Lee, and Omer Rana. The internet of things, fog and cloud continuum: Integration and challenges. *Internet of Things*, 3-4:134–155, 2018.

[22] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[23] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018.

[24] Chao Huang, Jianwei Huang, and Xin Liu. Cross-silo federated learning: Challenges and opportunities, 2022.

[25] Pete Beckman, Jack Dongarra, Nicola Ferrier, Geoffrey Fox, Terry Moore, Dan Reed, and Micah Beck. *Harnessing the Computing Continuum for Programming Our World*, pages 215–230. 2020.

[26] Qiang Duan, Shangguang Wang, and Nirwan Ansari. Convergence of networking and cloud/edge computing: Status, challenges, and opportunities. *IEEE Network*, 34(6):148–155, 2020.

[27] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018.

[28] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. *CoRR*, abs/2003.00295, 2020.

[29] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *CoRR*, abs/2102.07623, 2021.

[30] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. *CoRR*, abs/1910.07796, 2019.

[31] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378, 2019.

[32] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *CoRR*, abs/2007.07481, 2020.

[33] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *CoRR*, abs/2002.06440, 2020.

[34] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning: A meta-learning approach. *CoRR*, abs/2002.07948, 2020.

[35] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, and Jing Jiang. Fedproto: Federated prototype learning over heterogeneous devices. *CoRR*, abs/2105.00243, 2021.

[36] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *CoRR*, abs/1909.12488, 2019.

[37] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. *CoRR*, abs/2103.04628, 2021.

[38] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *CoRR*, abs/2003.13461, 2020.

[39] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Federated multi-task learning for competing constraints. *CoRR*, abs/2012.04221, 2020.

[40] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019.

[41] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints. *CoRR*, abs/1910.01991, 2019.

[42] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning, 2020.

[43] Yichen Ruan and Carlee Joe-Wong. Fedsoft: Soft clustered federated learning with proximal local updating, 2022.

[44] Manan Mehta and Chenhui Shao. A greedy agglomerative framework for clustered federated learning. *IEEE Transactions on Industrial Informatics*, pages 1–12, 2023.

[45] Donald Shenaj, Eros Fanì, Marco Toldo, Debora Caldarola, Antonio Tavera, Umberto Michieli, Marco Ciccone, Pietro Zanuttigh, and Barbara Caputo. Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 444–454, 2023.

[46] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. *CoRR*, abs/1804.08333, 2018.

[47] Fang Shi, Chunchao Hu, Weiwei Lin, Lisheng Fan, Tiansheng Huang, and Wentai Wu. Vfedcs: Optimizing client selection for volatile federated learning. *IEEE Internet of Things Journal*, 9(24):24995–25010, 2022.

[48] Sai Qian Zhang, Jieyu Lin, and Qi Zhang. A multi-agent reinforcement learning approach for efficient client selection in federated learning. *CoRR*, abs/2201.02932, 2022.

[49] Yuang Jiang, Shiqiang Wang, Bong Jun Ko, Wei-Han Lee, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *CoRR*, abs/1909.12326, 2019.

[50] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *CoRR*, abs/1811.11479, 2018.

[51] S. Yu, P. Nguyen, W. Abebe, W. Qian, A. Anwar, and A. Jannesari. Spatl: Salient parameter aggregation and transfer learning for heterogeneous federated learning. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, Los Alamitos, CA, USA, nov 2022. IEEE Computer Society.

[52] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. *CoRR*, abs/2007.07682, 2020.

[53] Ahmed M. Abdelmoniem and Marco Canini. Towards mitigating device heterogeneity in federated learning via adaptive model quantization. In *Proceedings of the 1st Workshop on Machine Learning and Systems*, EuroMLSys '21, page 96–103, New York, NY, USA, 2021. Association for Computing Machinery.

[54] Monik Raj Behera, Sudhir Upadhyay, Suresh Shetty, Sudha Priyadarshini, Palka Patel, and Ker Farn Lee. Fedsyn: Synthetic data generation using federated learning, 2022.

[55] Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyan Wu, Terrence Chen, David Doermann, and Arun Innanje. Preserving privacy in federated learning with ensemble cross-domain knowledge distillation. 2022.

[56] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization. *CoRR*, abs/1903.03934, 2019.

[57] Yujing Chen, Yue Ning, and Huzefa Rangwala. Asynchronous online federated learning for edge devices. *CoRR*, abs/1911.02134, 2019.

[58] Zhipin Gu and Yuexiang Yang. Detecting malicious model updates from federated learning on conditional variational autoencoder. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 671–680, 2021.

[59] Ying Zhao, Junjun Chen, Jiale Zhang, Di Wu, Michael Blumenstein, and Shui Yu. Detecting and mitigating poisoning attacks in federated learning using generative adversarial networks. *Concurr. Comput. Pract. Exp.*, 34(7), 2022.

[60] Melvin Chelli, Cédric Prigent, René Schubotz, Alexandru Costan, Gabriel Antoniu, Loïc Cudennec, and Philipp Slusallek. Fedguard: Selective parameter aggregation for poisoning attack mitigation in federated learning. In *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 72–81, 2023.

[61] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 2545–2555, New York, NY, USA, 2022. Association for Computing Machinery.

[62] Ashneet Khandpur Singh, Alberto Blanco-Justicia, and Josep Domingo-Ferrer. Fair detection of poisoning attacks in federated learning on non-i.i.d. data. *Data Mining and Knowledge Discovery*, Jan 2023.

[63] Yao Zhou, Jun Wu, Haixun Wang, and Jingrui He. Adversarial robustness through bias variance decomposition: A new perspective for federated learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 2753–2762, New York, NY, USA, 2022. Association for Computing Machinery.

[64] Mehdi Salehi Heydar Abad, Emre Ozfatura, Deniz Gündüz, and Özgür Erçetin. Hierarchical federated learning across heterogeneous cellular networks. *CoRR*, abs/1909.02362, 2019.

[65] Lumin Liu, Jun Zhang, S.H. Song, and Khaled B. Letaief. Client-edge-cloud hierarchical federated learning. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[66] Zhenpeng Liu, Sichen Duan, Shuo Wang, Yi Liu, and Xiaofei Li. Mflces: Multi-level federated edge learning algorithm based on client and edge server selection. *Electronics*, 12(12), 2023.

[67] Yongheng Deng, Feng Lyu, Ju Ren, Yongmin Zhang, Yuezhi Zhou, Yaoxue Zhang, and Yuanyuan Yang. Share: Shaping data distribution at edge for communication-efficient hierarchical federated learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 24–34, 2021.

[68] Shivam Kalra, Junfeng Wen, Jesse C. Cresswell, Maksims Volkovs, and Hamid R. Tizhoosh. Proxyfl: Decentralized federated learning through proxy model sharing. *CoRR*, abs/2111.11343, 2021.

[69] Rong Dai, Li Shen, Fengxiang He, Xinmei Tian, and Dacheng Tao. DisPFL: Towards communication-efficient personalized federated learning via decentralized sparse training. In *Proceedings of the 39th International Conference on Machine Learning*, pages 4587–4604. PMLR, 2022.

[70] Dong-Jun Han, Minseok Choi, Jungwuk Park, and Jaekyun Moon. Fedmes: Speeding up federated learning with multiple edge servers. *IEEE Journal on Selected Areas in Communications*, 39(12):3870–3885, 2021.

[71] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *CoRR*, abs/2103.00710, 2021.

[72] Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *CoRR*, abs/2003.02133, 2020.

[73] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *CoRR*, abs/1906.08935, 2019.

[74] Eugene Bagdasaryan and Vitaly Shmatikov. Differential privacy has disparate impact on model accuracy. *CoRR*, abs/1905.12101, 2019.

[75] Josep Domingo-Ferrer, David Sánchez, and Alberto Blanco-Justicia. The limits of differential privacy (and its misuse in data release and machine learning). *CoRR*, abs/2011.02352, 2020.

[76] Arnaud Grivet Sébert, Renaud Sirdey, Oana Stan, and Cédric Gouy-Pailler. Protecting data from all parties: Combining fhe and dp in federated learning, 2022.

[77] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *CoRR*, abs/1611.04482, 2016.

[78] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data, 2016.

[79] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. Analyzing federated learning through an adversarial lens. *CoRR*, abs/1811.12470, 2018.

[80] Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B. Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68:4583–4596, 2020.

[81] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning? *CoRR*, abs/1911.07963, 2019.

[82] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy, 2018.

[83] Batiste Le Bars, Aurélien Bellet, Marc Tommasi, Erick Lavoie, and Anne-Marie Kermarrec. Refined convergence and topology learning for decentralized sgd with heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 1672–1702. PMLR, 2023.

[84] TensorFlow Federated. https://github.com/tensorflow/federated, 2018.

[85] 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.

[86] G. Anthony Reina, Alexey Gruzdev, Patrick Foley, Olga Perepelkina, Mansi Sharma, Igor Davidyuk, Ilya Trushkin, Maksim Radionov, Aleksandr Mokrov, Dmitry Agapov, Jason Martin, Brandon Edwards, Micah J. Sheller, Sarthak Pati, Prakash Narayana Moorthy, Hans Shih-Han Wang, Prashant Shah, and Spyridon Bakas. Openfl: An open-source framework for federated learning. *CoRR*, abs/2105.06413, 2021.

[87] Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. Fate: An industrial grade platform for collaborative learning with data protection. *Journal of Machine Learning Research*, 22(226):1–6, 2021.

[88] Mathieu N. Galtier and Camille Marini. Substra: a framework for privacy-preserving, traceable and collaborative machine learning. *CoRR*, abs/1910.11567, 2019.

[89] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, et al. IBM federated learning: an enterprise framework white paper V0.1. *CoRR*, abs/2007.10987, 2020.

[90] Paddlepaddle/paddlefl: Federated deep learning in paddlepaddle. https://github.com/PaddlePaddle/PaddleFL.

[91] Holger R. Roth, Yan Cheng, Yuhong Wen, Isaac Yang, et al. Nvidia flare: Federated learning from simulation to real-world, 2022.

[92] Qinbin Li, Yanzheng Cai, Yuxuan Han, Ching Man Yung, Tianyuan Fu, and Bingsheng He. Fedtree: A fast, effective, and secure tree-based federated learning system. https://github.com/Xtra-Computing/FedTree/blob/main/FedTree_draft_paper.pdf, 2022.

[93] Facebookresearch. Flsim. https://github.com/facebookresearch/FLSim.

[94] Harshit Daga, Jaemin Shin, Dhruv Garg, Ada Gavrilovska, Myungjin Lee, and Ramana Rao Kompella. Flame: Simplifying topology extension in federated learning. In *Proceedings of the 2023 ACM Symposium on Cloud Computing*, SoCC '23, 2023.

[95] P2p federated learning (p2pfl). https://github.com/pguijas/p2pfl.

[96] Jae Hun Ro, Ananda Theertha Suresh, and Ke Wu. Fedjax: Federated learning simulation with JAX. *CoRR*, abs/2108.02117, 2021.

[97] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. *arXiv preprint arXiv:2007.01154*, 2020.

[98] Fan Lai, Yinwei Dai, Xiangfeng Zhu, and Mosharaf Chowdhury. Fedscale: Benchmarking model and system performance of federated learning. *CoRR*, abs/2105.11367, 2021.

[99] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018.

[100] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. Fednlp: A research platform for federated learning in natural language processing. *CoRR*, abs/2104.08815, 2021.

[101] Sixu Hu, Yuan Li, Xu Liu, Qinbin Li, Zhaomin Wu, and Bingsheng He. The OARF benchmark suite: Characterization and implications for federated learning systems. *CoRR*, abs/2006.07856, 2020.

[102] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *CoRR*, abs/2102.02079, 2021.

[103] Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings, 2022.

[104] Xiaoyuan Liu, Tianneng Shi, Chulin Xie, Qinbin Li, Kangping Hu, Haoyu Kim, Xiaojun Xu, Bo Li, and Dawn Song. Unifed: A benchmark for federated learning frameworks, 2022.

[105] Ivan Kholod, Evgeny Yanaki, Dmitry Fomichev, Evgeniy Shalugin, Evgenia Novikova, Evgeny Filippov, and Mats Nordlund. Open-source federated learning frameworks for iot: A comparative review and analysis. *Sensors*, 21:167, 12 2020.

[106] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Fednas: Federated deep learning via neural architecture search. *CoRR*, abs/2004.08546, 2020.

[107] Yi Zhou, Parikshit Ram, Theodoros Salonidis, Nathalie Baracaldo, Horst Samulowitz, and Heiko Ludwig. Flora: Single-shot hyper-parameter optimization for federated learning. *CoRR*, abs/2112.08524, 2021.

[108] Huanle Zhang, Mi Zhang, Xin Liu, Prasant Mohapatra, and Michael DeLucia. Automatic tuning of federated learning hyper-parameters from system perspective. *CoRR*, abs/2110.03061, 2021.

[109] Cédric Prigent, Gabriel Antoniu, Alexandru Costan, and Loïc Cudennec. Supporting Efficient Workflow Deployment of Federated Learning Systems across the Computing Continuum. SC 2022 - Salon SuperComputing, November 2022. Poster.

[110] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *CoRR*, abs/1910.10252, 2019.

[111] Siddharth Divi, Yi-Shan Lin, Habiba Farrukh, and Z. Berkay Celik. New metrics to evaluate the performance and fairness of personalized federated learning. *CoRR*, abs/2107.13173, 2021.

[112] Hongtao Lv, Zhenzhe Zheng, Tie Luo, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, and Chengfei Lv. Data-free evaluation of user contributions in federated learning. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pages 1–8, 2021.

[113] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier C. van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, Sudeep Agarwal, Julien Freudiger, Andrew Byde, Abhishek Bhowmick, Gaurav Kapoor, Si Beaumont, Áine Cahill, Dominic Hughes, Omid Javidbakht, Fei Dong, Rehan Rishi, and Stanley Hung. Federated evaluation and tuning for on-device personalization: System design & applications. *CoRR*, abs/2102.08503, 2021.

[114] Waris Gill, Ali Anwar, and Muhammad Ali Gulzar. Provfl: Client-driven interpretability of global model predictions in federated learning, 2023.

[115] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[116] neptune.ai. https://neptune.ai/.

[117] Experiment tracking with weights and biases. https://www.wandb.com/.

[118] Matei A. Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41:39–45, 2018.

[119] Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning. *CoRR*, abs/2007.13518, 2020.

[120] Influxdata/telegraf: The plugin-driven server agent for collecting & reporting metrics. https://github.com/influxdata/telegraf.

[121] Influxdata/influxdb: Scalable datastore for metrics, events, and real-time analytics. https://github.com/influxdata/influxdb.

[122] Grafana/grafana: The open and composable observability and data visualization platform. visualize metrics, logs, and traces from multiple sources like prometheus, loki, elasticsearch, influxdb, postgres and many more. https://github.com/grafana/grafana.

[123] D. Rosendo, M. Mattoso, A. Costan, R. Souza, D. Pina, P. Valduriez, and G. Antoniu. Provlight: Efficient workflow provenance capture on the edge-to-cloud continuum. In *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 221–233, Los Alamitos, CA, USA, nov 2023. IEEE Computer Society.

[124] Amazon sagemaker. https://aws.amazon.com/sagemaker/.

[125] Kubeflow/kubeflow: Machine learning toolkit for kubernetes. https://github.com/kubeflow/kubeflow.

[126] Daniel Rosendo, Pedro Silva, Matthieu Simonin, Alexandru Costan, and Gabriel Antoniu. E2Clab: Exploring the Computing Continuum through Repeatable, Replicable and Reproducible Edge-to-Cloud Experiments. In *Cluster 2020 - IEEE International Conference on Cluster Computing*, pages 1–11, Kobe, Japan, September 2020.

[127] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, et al. Adding virtualization capabilities to the Grid'5000 testbed. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.

[128] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and Thomas Watteyne. Fit iot-lab: A large scale open experimental iot testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 459–464, 2015.

[129] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. Lessons learned from the chameleon testbed. In *Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC'20, USA, 2020. USENIX Association.

[130] Etp4hpc strategic research agenda. https://www.etp4hpc.eu/sra.html.

[131] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[132] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B. Gibbons. Federated learning under distributed concept drift, 2023.

[133] Gartner keynote: The future of business is composable. https://www.gartner.com/smarterwithgartner/gartner-keynote-the-future-of-business-is-composable. [Online; accessed 23-January-2023].

[134] Vaishak Belle and Ioannis Papantonis. Principles and practice of explainable machine learning. *Frontiers in Big Data*, 4, 2021.

[135] Mohammad Mohammadi Amiri, Deniz Gündüz, Sanjeev R. Kulkarni, and H. Vincent Poor. Federated learning with quantized global model updates. *CoRR*, abs/2006.10672, 2020.

[136] Ousmane Touat and Sara Bouchenak. Towards robust and bias-free federated learning. In *Proceedings of the 3rd Workshop on Machine Learning and Systems*, EuroMLSys '23, page 49–55, New York, NY, USA, 2023. Association for Computing Machinery.