Decentralized Learning in Federated Deployment Environments: a System-level Survey

PAOLO BELLAVISTA, LUCA FOSCHINI, and ALESSIO MORA, Dept. Computer Science and Engineering (DISI), Alma Mater Studiorum - University of Bologna

Decentralized learning is attracting more and more interest because it embodies the principles of data minimization and focused data collection, while favouring the transparency of purpose specification (i.e. the objective a model is built for). Cloud-centric-only processing and deep learning are no longer a strict necessity to train high-fidelity models; edge devices can actively participate in the decentralized learning process by exchanging meta-level information in place of raw data, thus paving the way for better privacy guarantees. In addition, these new possibilities can relieve the network backbone from unnecessary data transfer and allow to meet strict low-latency requirements by leveraging on-device model inference. This survey provides a detailed and up-to-date overview of the most recent contributions available in the state-of-the-art decentralized learning literature. In particular, it originally provides the reader audience with a clear presentation of the peculiarities of federated settings, with a novel taxonomy of decentralized learning approaches, and with a detailed description of the most relevant and specific system-level contributions of the surveyed solutions for privacy, communication efficiency, non-IIDness, device heterogeneity, and poisoning defense.

CCS Concepts: • **Computing methodologies → Distributed computing methodologies**; **Distributed algorithms**; **Distributed artificial intelligence**; **Learning settings**.

Additional Key Words and Phrases: Decentralized Learning, Federated Deployment, Privacy, Communication Efficiency, Poisoning Defense

ACM Reference Format:
Paolo Bellavista, Luca Foschini, and Alessio Mora. 2018. Decentralized Learning in Federated Deployment Environments: a System-level Survey. 1, 1 (December 2018), 38 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The unprecedented amount of data being generated at the edge of the network — Cisco estimates that nearly 850 ZB will be produced by all, namely, people, machines, and things by 2021, up from 220 ZB generated in 2016 [21] — represents the ideal ingredient for training accurate Machine Learning (ML). In particular, Deep Learning (DL) models [63] allow to enhance and support a wide range of more intelligent applications, services, and infrastructures, such as powering recommender systems [139], developing data-driven machine health monitoring [143], enabling new ways for clinical diagnoses [86], or driving the design of new generation mobile networks [137]. However, the potentially sensitive or confidential nature of gathered data poses privacy concerns when managing, storing, and processing those data in centralized locations. At the same time, the capacity of the

Authors' address: Paolo Bellavista, paolo.bellavista@unibo.it; Luca Foschini, luca.foschini@unibo.it; Alessio Mora, alessio.mora@unibo.it, Dept. Computer Science and Engineering (DISI), Alma Mater Studiorum - University of Bologna, Viale Risorgimento 2, Bologna, Italy, 40136.

XXXX-XXXX/2018/12-ART $15.00
https://doi.org/10.1145/1122445.1122456

network infrastructure risks to be saturated by such continuous data collection, such as from distributed sources at the network edge to centralized cloud resources.

To this purpose, decentralized learning has recently gained momentum exactly to decouple model training from the need of directly accessing raw data, by becoming a promising alternative solution to the more traditional cloud-based ML. In fact, decentralized learning leaves the training data distributed and supports the learning of joint models via local computation and periodic communication: data no longer need to leave the data owner. For example, data remain on the premises of organizations or institutions that may want to collaborate, but without sharing their private data. Other significant use cases embrace intelligent applications for end-users of smartphones or IoT devices, where the private preferences or habits sensed through user-device interaction do not leave the source devices.

The literature includes several differently designed approaches to enable decentralized learning. The common key idea is to be able to just transmit ephemeral locally-computed updates (e.g., model parameters or gradients) and/or meta-level information (e.g., activations in neural-networks): that leverages on the fact that they are meaningful only with respect to the current global model and typically bring significantly lower informative content compared to the raw data (data processing inequality). This design paves the way to upgrading the user's privacy so to meet the rising legislative requirements about it (e.g., the California Consumer Privacy Act [93] and the European General Data Protection Regulation (GDPR) [30]). Similarly, in the case of federated deployment environments participated by different institutions, the use of decentralized learning techniques can ensure privacy guarantees, especially in sensitive domains such as healthcare where data sharing is impeded by regulation (e.g., the Health Insurance Portability and Accountability Act - HIPAA [94]).

Besides the above privacy concerns, decentralized learning techniques are strongly motivated from the infrastructural perspective. The huge amount of raw data coming from the edge of the network and headed to datacenters risks to overwhelm the network backbone, hence a part of these data should, instead, be consumed locally, as suggested in [21]. Note that, even with decentralized learning, the periodic exchange of uncompressed updates in place of the upload of all the raw data may not necessarily reduce the total communication cost needed to train a model in a satisfying way [76].

As for the paper organization, this survey firstly presents the motivations that led to the development of decentralized learning and provides a practical overview about its real-world applications (in Section 2). Then, it defines the peculiarities of federated deployment environments (or federated settings in Section 3.1), introduces our original taxonomy to classify decentralized learning approaches, and presents the main baselines for enabling decentralized learning (in Section 3). In Section 4, it points out the main issues that have been addressed by the related literature in the last four years. Indeed, that represents the core of our work providing an accurate, but largely accessible, overview of the major works in the current literature about decentralized learning. The referred works are readily characterized in the first place by the federated setting they refer to (i.e., Cross-silo or Cross-device), second, by a simple modular description of the baseline framework on which the particular work is based (using our taxonomy from Section 3.2), and third by the specific issues addressed in the surveyed solutions (i.e., privacy, communication efficiency, non-IIDness, device heterogeneity, poisoning defense). The last part of this survey (in Section 5) looks at present and future research directions for the advancement of decentralized learning, by discussing open technical challenges and cutting edge lines of work.

We are aware of the rich existing survey literature in the field and in particular of the valuable [68], [129], [73], and [147] papers. However, we claim that we are providing the readers with a valuable and differentiated contribution if compared with those surveys primarily because of the following aspects:

(1) We provide a more in-depth and more extensive technical description of the surveyed works, describing their motivations, bringing out their most significant technical insights, and providing the readers with the references to fully comprehend the associated solution guidelines, as well as commenting their differential strengths and weaknesses.
(2) We provide a readily and intuitive characterization of the surveyed works by means of a tabular road map to approach the core of our survey, and we claim that it may be useful to help non-expert readers to navigate the very differentiated literature that is emerging in the field.
(3) Our survey includes several very recent research papers (published in the last few months) that are relevant for the community and not covered yet by [68] and [129].
(4) We enlarge the discussion to cover decentralized learning approaches in a broader sense, not focusing exclusively on federated learning related works.
(5) Finally, differently from [73] and [147], we do not specifically focus only on the advances of decentralized learning that can be achieved via Multi-access Edge Computing (MEC).

2 THE RISING OF DECENTRALIZED LEARNING

The public opinion is becoming increasingly sensitive to individual privacy rights, especially after the notorious Facebook-Cambridge Analitica scandal [126] has made no longer ignorable the Orwellian levels of data held by such companies about us and has exposed the weakness (or even the non-existence) of privacy regulation and data protection. Anyway, even without thinking to striking episodes such the above cited one, individuals' privacy is threatened whenever personal raw data are disclosed. For example, elementary data anonymization (i.e., removing all explicit identifiers such as name, address, and phone number) has demonstrated to be almost ineffective in protecting privacy, since combinations of simple non-unique attributes often allow to re-identify individuals by matching "anonymized" records with non-anonymized ones in a different public dataset (e.g., [88]).

The actual legislative vacuum about data harvesting, data holding, and data processing has been — and still is — the subject of regulation efforts around the world. About that, it is worth mentioning the CCPA and the GDPR, respectively from California and from European Union, that both leverage the principles of *purpose specification* and *data minimization*. In concrete terms, for example, the GDPR's Article 5 states that personal data shall be "collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes" and "kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed". Such guidelines are often incompatible with more traditional cloud-based ML solutions, where potential privacy-sensitive raw data flow towards datacenters to train ML/DL models. In particular, *(i)* companies harvesting data tend to keep them forever and users cannot delete them[1], hence same data can be used several times for different learning purposes (for extracting different kinds of insights); *(ii)* users from whom the data were collected are unaware of the associated learning objectives; *(iii)* models learnt from collective data typically remain property of the companies that built them; and *(iv)* users disclose their raw data, in a more or less informed way, to infer centralized models, such as for training.

It could seem that an inevitable dichotomy between the protection of individual's privacy and the distillation of useful knowledge from a population exists (i.e., not disclosing private data to preserve privacy, by merely performing local learning, versus sharing private raw data to produce more accurate models at the cost of exposing data owners to privacy violation risks). On the opposite, decentralized learning tries to alleviate the privacy concerns of traditional cloud-centric

[1] At least until the time this survey has been written.

training by design and is data-minimization-prone. In fact, *(i)* companies do not need anymore to collect possible privacy-sensitive raw data to build ML/DL models; *(ii)* users could likewise be unaware of the learning objective for which their data are used, but data processing happens locally, hence facilitating the shift to full transparency; *(iii)* models (or fractions of models, i.e., portions of their parameters) reside locally at the user's device or inside the organization's premises (or in very proximity of it). This could be seen as a first step to give back to the community the knowledge acquired from joint contributions[2]; *(iv)* users do not need to upload their raw data to query centralized models, in fact on-device inference is typically enabled if the entire model is replicated locally — if only a portion of the model parameters is locally held instead, distributed inference is performed by just communicating meta-level information in place of raw data.

In addition, shifting model training from the cloud towards the network edge recalls a trend that was already in act with the rising of mobile edge computing during the last decade. Besides the urge of privacy guarantee, several aspects are similar and seem to overlap. A primary one is the need to relief the burden on the backbone of the network infrastructure, which risks to collapse under the tsunami of data if not partially consumed locally or in proximity of the associated sources. Intuitively, actively involving the ecosystem of edge devices in the learning process and exchanging model updates in a communication-efficient way (e.g., employing stream compression) in place of centralizing raw data can substantially reduce network traffic while leading to limited degradation (or in some cases to no degradation) of model accuracy. Secondly, the low-latency requirements of real-time applications often cannot be met by only leveraging the cloud (for instance when monitoring a shared industrial workspace, during human robot collaboration, to enforce policies for worker protection [108]). Enabling on-device inference of the learned or in-learning models, which naturally comes with most decentralized learning approaches as we will discuss in the continuation of the survey, benefits such delicate aspect. Let us finally note that decentralized training, with its potential reduction of ML-related energy consumption because of reduced network traffic and decreased transmission distance, also contributes to the overall sustainability of the approach: it is considered as one of the key enabling technologies towards green networking via distributed and federated datacenters.

Decentralized learning finds natural applications in smart apps for mobile devices which learn by user interaction, and where low-latency responses are required. In this context, gathering user-labeled or automatically annotated data points for feeding supervised learning algorithms is a common practice. Related examples include on-device intelligent keyboards that power content suggestions [130], or that predict the most suitable next words [38] or the most fitting Emojis [100] given the chat history; or again vocabularies that evolve to follow the ongoing trending expressions by learning out-of-vocabulary words [18], and all of this without exporting sensitive text to servers. Other examples deal with human activity recognition (e.g., [113]) and keyword spotting for voice assistants in smart homes (e.g., [64]) .

Decentralized learning has been used also to conjugate user privacy and prediction ability of the infrastructure in the 5G multi-access edge computing architecture [57] [24] [80], for example for proactive content caching [135] or for optimal allocation of virtual machine replicas copies [31], and it is considered a key enabling tool for next generation wireless networks [90] as well, e.g., for spectrum management.

Confirming its versatility, decentralized learning has been also applied to network traffic classification, anomaly detection, and VPN traffic recognition tasks, while preserving appropriate privacy

[2]However, it is worth noting that restricting or preventing access to model's parameters, even if the model itself is locally available, makes it harder for an attacker to undermine it, e.g., via backdooring. Therefore, companies or organizations that adopt Decentralized Learning techniques may be anyway motivated to hamper model inspection.

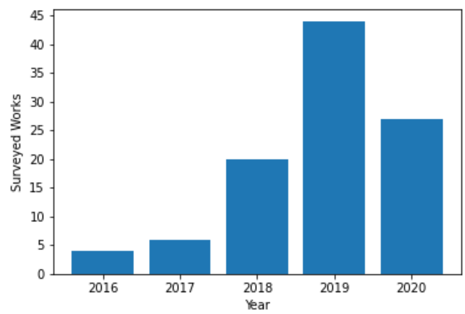

Fig. 1. The histogram reports the number of papers about decentralized learning per year, covered by this survey, by showing the increasing relevance of decentralized learning in the literature.

levels [144] [8]. Similar considerations apply to vision-based safety monitoring systems in smart cities [78].

In the relevant healthcare domain, the popularity of decentralized training approaches shown in Figure 1 has been also pushed by the need to enable collaboration among healthcare institutions. In fact, the disclosure of patients' raw data is often impeded or limited by regulations such as the HIPAA Privacy Rule, or the patient herself might not want her clinical data to be released to other entities, or again the institutions might not want to sell out their valuable datasets. Therefore, plain old centralized training results to be not feasible for predictive clinical models in many cases. Furthermore, manual labeling of data is often very time-consuming in medical contexts and typically requires qualified personnel. Datasets held by single institutions tend to be small and may lack in diversity [95], and this is exacerbated when considering rare diseases. Hence, from the perspective of isolated local learning, sample scarcity may lead to models with poor predictive ability, especially when considering deep learning models that notoriously need abundant data points to reach high fidelity. As practical use cases in smart healthcare, we report the training of a detector for abnormal retinal fundus and a classifier for common chest radiography observations (from visual datasets) [99]. Other clinical learning tasks include prediction of prolonged length of stay and in-hospital mortality [96], prediction of hospitalizations for cardiac events [15], or gaining insights about brain diseases [104].

3 FUNDAMENTALS, TAXONOMY AND BASELINES FOR DECENTRALIZED LEARNING

This Section gives some concise background to make highly accessible the following presentation of the surveyed decentralized learning solutions, by defining the targeted deployment settings and the modular building blocks that are emerging in the related literature. These building blocks are at the cornerstones of our original taxonomy, which we will introduce in this Section and use in the remainder of the survey to better highlight the features, the pros, and the cons of the surveyed contributions. We also present the most interesting baseline solutions to enable decentralized learning.

3.1 Cross-Silo and Cross-Device Federated Settings

Here we provide an informal and qualitative characterization of the two most common settings for decentralized learning, by highlighting their specific elements with respect to traditional distributed settings [22]. As anticipated in the previous sections, decentralized learning techniques are strongly motivated when data sharing is impeded by law or by privacy concerns, hence they apply to several real-world contexts. For the sake of simplicity, let us consider two extreme scenarios: *(i)* the federation of entities participating in collaborative learning tasks consists of compute nodes from different organizations or companies (e.g., hospitals, banks) — that typically store their private data in on-premise silos —; *(ii)* the federation comprises a massive amount of edge devices (such as smartphones, IoT devices, or IIoT devices). Such primary distinction leads to the identification of two very general settings, which we respectively name Cross-silo federated settings and Cross-device federated settings [53].

Those two federated scenarios are substantially different from more traditional distributed settings, where raw data are centralized in datacenters to perform learning. In fact, in cloud-centric training, the participants of the learning task are compute nodes (generally up to 1000) interconnected through very fast networks, making the computation cost the major bottleneck. Data can be balanced across compute nodes; moreover, they can be partitioned and re-partitioned according to the need. Importantly, any participant can access any part of the dataset. Worker machines are reliable and low rate of failure or drop out (i.e., abandoning the learning task without notice) are expected.

The Cross-silo federated setting refers to a scenario in which the entities involved in the learning process are limited in number (up to 100 participants), and typically they are trusted and reliable. In addition, they are likely to participate in the entire training task. Data can be unbalanced, but in general not as much as in Cross-device settings. No assumptions about communication or computation bottlenecks are made a priori. Furthermore, while training data are assumed to be independently and identically distributed (IID) in typical datacenter settings, such assumption does not hold for federated settings (neither for Cross-silo nor for Cross-device): the training data on a given device or on a given machine are likely not to be representative of the full population distribution.

In the Cross-device federated settings, participants are very numerous instead (up to 10^{10}), data are massively distributed and unbalanced (e.g., the number of training examples held by participants can differ by one or two orders of magnitude) [60]. Learners are highly unreliable; failure and drop out must be addressed, and each client is likely not to take part in the entire training process (actually they may contribute only once per task). Furthermore, since edge devices have limited bandwidth, communication efficient solutions are preferable in Cross-device setting; the federation may comprise computationally constrained devices as well, making more delicate the computation/communication trade-off. Another peculiarity is that participants may be malicious in this scenario, e.g. trying to infer sensitive information about other learners or voluntarily hampering the global learning.

For the sake of clarity, we use this characterization[3] to readily approximate the setting to which the surveyed works in Section 4 refer — we will show that the targeted federated setting relevantly influences the design choices of a solution. We indeed use such characterization of the setting as a primary dimension of our taxonomy.

3.2 A Taxonomy for Decentralized Learning Systems

To favour the readability of the remainder of the survey, we propose a taxonomy for decentralized learning systems that highlights the main alternative options in designing such frameworks.

[3] We use the terminology found in [53]. However, the existence of a central orchestrator (i.e., an entity orchestrating the collaborative training) in federated settings, either Cross-silo or Cross-device, is further supposed in [53]. To embrace all the decentralized learning work from the literature, we relax this last trait in our terminology usage in this paper.

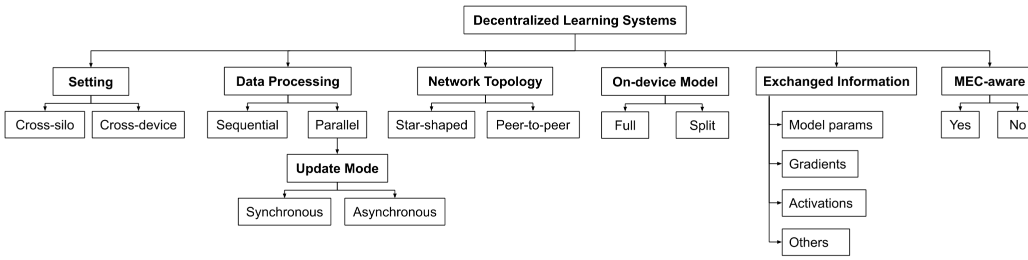

Fig. 2. Our taxonomy for decentralized learning systems.

3.2.1 Data processing: Data-sequential vs Data-parallel. The common thread when designing decentralized learning algorithm is leveraging data-parallel variants of iterative optimization algorithms that are inherently sequential, e.g. Stochastic Gradient Descent (SGD) and its optimizations. Typically, the federation of learners collaborates to minimize a global objective function, that is unknown to the participants since no single node has direct access to all the data. The global objective can be thought as a linear combination of the local empirical losses, available locally to the participants [60].

We further divide data-parallel approaches into systems that leverage *synchronous* or *asynchronous update mode*. In fact, as traditional distributed training algorithms, also data-parallel decentralized learning approaches can exploit asynchronous updates to optimize on speed by using potentially stale parameters for local training or wait for local computation of the slowest participant to synchronously aggregate updates without risking to use outdated parameters. With synchronous update mode, it is usual to talk about rounds of communication, i.e., all the triggered participants retrieve the global model state, produce their locally computed updates and communicate such updates, from which the new generation model will be derived. Communication efficient algorithms have their principal goal in minimizing the rounds of communication. Relaxing the synchronicity can instead spread the communications over time, particularly helpful when handling a large number of learners. However, examples of data-sequential systems exist, i.e., systems in which each participant uses as starting model state the result of the computation of another participant, and thus produces as output the input model state for the next participant. Anyway, let us note that these solutions are usually limited to the Cross-silo setting.

3.2.2 Network Topology: Star-shaped vs Peer-to-peer. The coordination among learners can be facilitated by a star-shaped network topology that leverages a central entity to distribute the current state of the global model at the beginning of each local iteration, and maintain the state updated during the training task. Participants can directly exchange their locally computed updates as well, in a peer-to-peer fashion, hence not requiring any infrastructure at the price of increased coordination complexity. In literature, decentralized learning frameworks that exploit peer-to-peer networks of participants are often referred as fully decentralized, i.e., decentralized in both data and coordination.

3.2.3 On-device Model: Full Model vs Splitted Model. Besides the full local replication of the (current) global model during the training process, it can be possible to have participants that are only responsible for a fixed subset of model parameters (in this case, typically, the parameters belonging to n shallower layers in a deep neural network, i.e. splitted models). The full replica of the global model enables on-device inference by design, while in the case of splitted model, without retrieving the entire model at the end of the training, distributed inference is required. Note that, anyway, the primary privacy concerns have been bypassed by having feature extraction locally[4].

3.2.4 Exchanged parameters: Model Parameters, Gradients, Activations and Others. We also emphasize that the degrees of freedom in designing decentralized learning frameworks also involve the kind of exchanged information during the distributed learning. Supposing gradient descent based methods for optimization, the usual practice is to have participants exchanging gradients or model updates, with the latter option valuable in case of participant-specific local solver. In star-shaped topology, a common practice is to have participants downloading the current model parameters and communicating back to the aggregator either the gradients or the locally updated model parameters typically generated through SGD iteration(s). Hence, with such topology it is usual to talk about parameters in upload and in download. There are examples of star-shaped frameworks where the communication in both the directions only involves gradient information (e.g., [118], [9]) as well, i.e., the server aggregates gradients and the back-propagation is performed on-device. We underline that the exchanged information may be not limited to gradients and model parameters, in fact other kinds of parameters may be transmitted for diverse optimization purposes. For instance, the exchange of moment estimates to implement an ADAM[59]-inspired optimization algorithm [85], or also of information for gradient correction terms [70], and of control variates [56] to tackle non-IIDness, or of other local estimations to meet given budget resources [125] (more details about their motivations and implementations are in Section 4). Or again, in presence of splitted models (e.g., in Split Learning), besides model parameters and gradients, also activations (and labels) have to be communicated by design.

3.2.5 MEC-awareness: Yes/No. It is also worth mentioning that, considering the MEC architecture and therefore the existence of a middle layer of edge servers between the edge devices and the cloud, two levels of topology organization can be identified. On the one hand, decentralized learning systems may leverage edge servers as intermediate aggregators for updates produced by the edge devices in their locality (i.e., matching a star-shaped topology) and then edge servers may directly exchange intermediate-level updates among them in a peer-to-peer fashion, to collaboratively build the global model. On the other hand, the cloud may be involved as "master aggregator" collecting intermediate aggregations from the federation of edge servers (the latter solution is referred as *hierarchical*). An in-depth discussion about edge-cloud continuum roles in edge intelligence can be found in [147].

3.3 Baselines for Decentralized Learning Systems

In this subsection, we propose some baseline frameworks to enable decentralized learning. We introduce the most significant baselines for star-shaped systems, followed by instances of fully decentralized (server-less) alternatives, i.e. peer-to-peer.

3.3.1 Star-shaped Baselines. Federated Averaging (FedAvg) is a widely accepted heuristic algorithm used as baseline for star-shaped Federated Learning (FL), given its simplicity and its empirical effectiveness [81] also in non-convex setting. Its skeleton is presented in Algorithm 1. The learning process proceeds in synchronous rounds of communication; the (full) current global model is broadcasted at the beginning of the round to the (selected) participants, that use their private dataset to produce an update (e.g., gradients or model weights) for the received model, and upload such contributions. The aggregator, i.e. a sort of parameter server, collects and aggregates (e.g., by averaging) the updates from participants and computes the new-generation global model. The process typically ends when a certain accuracy for the global model is reached, or when a certain

[4]It is important to remind that information leakage is still possible. This will be faced in Section 4.2.

Algorithm 1: FedAvg algorithm
The K participants are indexed by k, $\mathcal{D}_k$ is the local dataset at participant k, $n_k = |\mathcal{D}_k|$ and $n = \sum_{k=1}^{K} n_k$, B is the local minibatch size, E represents the number of local epochs, η is the learning rate. Note the common initialization of model parameters w_0.

```
Server executes:
    initialize w_0
    for each round t = 1, 2, 3, ..
        m ← max(C × K, 1)
        S_t ← (random set of m clients)
        for each client k ∈ S_t in parallel
            w^k_{t+1} ← ClientUpdate(k, w_t)
        w_{t+1} ← Σ_{k=1}^{K} (n_k / n) w^k_{t+1}
ClientUpdate(k, w)
    𝓑 ← (split 𝒟_k into batches of size B)
    for each local epoch e from 1 to E
        for batch b ∈ 𝓑
            w ← w − η∇ℓ(w; b)
    return w to server
```

number of rounds has been executed. SGD is typically chosen as local solver. Three hyperparameters have to be tuned in FedAvg; C controls the fraction of participants to be selected in a certain round t (with $C = 0.0$ indicating only one participant involved per round, and $C = 1.0$ meaning the totality of participants), E defines the number of local epochs to be performed in each round, and B denotes the minibatch size. It is worth noting that the contributions in the aggregation are weighed accordingly to the number of local data points held by each participant.

When the full local dataset is treated as a single minibatch (i.e., $B = \infty$), and the local iterations at each participant are limited to one epoch (i.e., $E = 1$), FedAvg is also known as FedSGD. An equivalent variant of FedSGD can be formulated by uploading gradients in place of model parameters.

An accurate convergence analysis, in strongly convex and smooth problems, of FedAvg in presence of data heterogeneity and partial device participation — peculiar of cross-device settings — can be found in [71]. The authors theoretically showed that, in such circumstances, model convergence is slowed down with respect to the ideal case of IIDness and full participation. They also pointed out that a decaying learning rate is fundamental for the convergence of FedAvg under non-IIDness: gradually diminishing the learning rate can neutralize biased local updates. Considering FL-suitable participant sampling and related averaging schemes, the authors of [71] establish a convergence rate of $\mathcal{O}(\frac{1}{T})$, where T represents the total number of SGD iterations performed by every participant.

FedAvg is considered a communication efficient algorithm mainly thanks to two aspects: (i) it selects a (random) subset of participants per round (i.e., if only a portion of participants is selected, the per-round communication cost is reduced with respect to full participation); (ii) it allows for additional iterations of local solver (i.e., SGD) to reduce the total number of synchronizations needed for model convergence – it has been empirically showed that FedAvg significantly reduces the total communication rounds (under the same C-fraction of per-round selected clients) with respect to FedSGD, while reaching the same (or higher) model accuracy [81]. A plethora of works in literature propose improvements for FedAvg (see Section 4 for further details).

A baseline alternative to FedAvg, Federated Distillation (FD), is presented in [49], and it is explicitly designed to be extremely communication efficient; it is inspired by an online version of knowledge distillation, namely co-distillation [44], [4]. In a nutshell, each device (the student) stores its model outputs, i.e. a set of logit values normalized via softmax function, from which it derives per-label mean logit vectors, and periodically uploads such local-average logit vectors to the aggregator. The server produces the per-label global-average logit vector by averaging the contributions of all the participants in that round, and broadcasts such aggregation to the federation; each device treats the received per-label global-average logit vector as the teacher's output, and locally calculates the distillation regularizer. It is straightforward to note that exchanging logit-vector (local or global averaged, whether they are upload or download parameters), in place of model parameters or gradients, reduces the per-round communication cost with respect to FedAvg: the dimension of logit-vectors depends on the number of labels, and not on the number of model parameters.

A differently designed method to enable collaborative training of neural networks without sharing raw private data is the so-called Split Learning (SL), also referred as SplitNN [36] to emphasize the suitability for DL architectures. This technique employs *splitted models* instead of *full model replication*. In fact, the training participants hold replications of the shallower layers up to a certain layer (i.e., the *cut layer*), and a central entity holds the deeper layers. Inter-layer values, i.e., activations and gradients exchange occurs between a certain participant and the central entity, instead of centralizing the raw data.

The training process as formulated in [36] is data-sequential, albeit distributed. Each participant retrieves the current state of the shallower layers of the neural network either in a peer-to-peer mode, downloading it from the last training participant, or in a centralized mode, downloading it from the central entity itself, and runs the local gradient descent based local solver (e.g., SGD), using its private dataset[5]. The participant computes the forward propagation up to the *cut layer*, and the outputs of this layer, together with label associated to the data examples, are communicated to the central entity that concludes the forward pass on the deeper layers. The back propagation of gradients takes place in a similar fashion, flowing from the deepest layer to the cut layer, where they are sent from the central entity to the participant that has initially triggered the forward propagation (only the gradients that refers to the *cut layer*). Then, the process repeats with a different participant, collectively learning a joint model without sharing private raw data. In [111] the position of the *cut layer* is empirically discussed.

Authors of [36] also proposed a variant of the SplitNN algorithm, namely U-shaped Split Learning, in which the labels related to the locally available training examples are not centralized but remains private at the participant side.

A data-parallel variant of SplitNN is proposed in [119], namely SplitFed learning (SFL), to combine the advantages of FL and SL, that are respectively the parallel processing among distributed learners and the model partitioning among participants and central entity.

Although splitNN has demonstrated to reduce computation burden and bandwidth utilization with respect to baseline FedAvg [111] in presence of "big" models and high number of clients, star-shaped FL and fully decentralized FL allow on-device inference of the model by design, while this is not true for splitNN that requires a distributed inference unless the complete trained model is provided to the participants.

3.3.2 Peer-to-peer baselines. In star-shaped FL, the coordination server orchestrates the communication rounds; it iteratively broadcasts the current model state to the participants and gathers the locally computed updates to produce the next-generation model by aggregation. Although leveraging a client-server architecture permits to ignore topology-related issues, FL presents two

[5]Regardless of the strategy to retrieve the current state of the participant-side model, either peer-to-peer or centralized, in SplitNN a server exists by design; this is why we consider it as star-shaped.

downsides: (i) the central entity can be seen as a single point of failure; (ii) the central entity may represent a bottleneck considering a significant number of training participants (as demonstrated in [72] though not explicitly targeting federated settings). Furthermore, the learners should trust such central aggregator, and, even though techniques such as multi-party computation can ensure inscrutability of updates (see Section 4.2), the participants may prefer to coordinate each others directly (as could be the case of health institutions).

In fully decentralized learning, the topology of star-shaped FL becomes a peer-to-peer topology, represented as a connected graph (generally assumed to be sparse). Such graph can be a directed graph or an undirected graph, i.e. unidirectional or bidirectional channels of communication among the nodes. The topology can be assumed to be fixed or dynamic, i.e. in which interconnections between nodes may change over time.

In each round, participants perform local computation and then communicate with (a subset of) the other nodes in the graph — note that not leveraging the server-client architecture (as well as relaxing the synchronous update mode) redefines the semantic of *rounds*. Straightforward optimization algorithms, similarly to FedAvg, employ fully decentralized variants of SGD (e.g., peers directly exchanging and merging gradients or model updates). It is also worth highlighting that, while in star-shaped FL the FedAvg algorithm has been widely accepted as baseline, in peer-to-peer (server-less) FL there is no algorithm that has distinctly emerged among others; solutions in literature, in fact, make different assumptions on the connectivity of the graph, in particular considering each node connected to all the other nodes in the network or considering only a set of nodes (i.e., the neighbours) reachable by each one, considering a fixed topology or a dynamic topology, assuming directed (e.g., [42]) or undirected graphs, and employing different strategies for model fusions.

In the continuation of this subsection, we present examples of baseline algorithms that consider fixed-topology and undirected graphs — most common assumptions. The first work, BrainTorrent [104], targets cross-silo federated settings, while the subsequently presented ones also embrace the cross-device setting [43] [50] [108].

BrainTorrent considers the graph as fully connected, from this consideration comes our labeling as cross-silo framework — it explicitly targets the collaboration of medical institutions, where it is reasonable to further suppose full connectivity besides fixed topology and undirected network graph. In a nutshell, a random participant k in the network starts the learning process by pinging all the others node requesting for model updates; the ones that have a fresher version of the model respond with their model parameters; the learner that has initiated the process, gathers the updates from the subset of participants that have responded, referred as $N_{\bar{k}}$, and aggregates them with its own local model by using this strategy: $\psi^k = \frac{n_k}{n} w^k + \sum_{i \in N_{\bar{k}}} \frac{n_i}{n} w^i$. Next, the participant k fine tunes the aggregated model ψ^k using its own private dataset, it updates the version of its model and it is ready to respond to ping request from other nodes by providing its new generation fine-tuned w_k. Then the process repeats.

Gossip-based protocol for distributed learning has been explored in the datacenter setting as alternative to the parameter-server approach (e.g., [10], [39]). Inspired from them, Gossip Learning (GL) has been proposed in [43] for Cross-device federated settings. In the baseline GL algorithm, starting from a common initialization, each node sends its local model to a randomly selected peer, which firstly merges (e.g., by averaging and weighing the average according to an age parameter associated with the freshness of the models) the received model with its current parameters, then updates the resulting model by exploiting its private dataset, and the process repeats. In a nutshell, there could be different models scattered across the network of peers, with each one of these models taking random walks (in the network) and being updated when visiting a new node. Typically, the

Algorithm 2: Consensus FedAvg algorithm
$N_{\overline{k}}$ represents the set of neighbors of the participant k, hence k excluded, $\mathcal{D}_k$ is the local dataset at participant k, B is the local minibatch size, η is the learning rate.

```
Participant k executes:
    initialize w_0^k
    for each round t = 1, 2, 3, ..
        receive{w_t^i}_{i ∈ N_k̄}
        ψ_t^k ← w_t^k
            for all devices i ∈ N_k̄
            ψ_t^k ← ψ_t^k + ζ_t α_{t,i}(w_t^i − w_t^k)
        w_{t+1}^k = ModelUpdate(ψ_t^k)
        send(w_{t+1}^k) to neighbors

ModelUpdate(ψ_t^k)
    𝓑 ← (split 𝒟_k into batches of size B)
    for batch b ∈ 𝓑
        ψ_t^k ← ψ_t^k − η∇ℓ(ψ_t^k; b)
    w_t^k ← ψ_t^k
    return(w_t^k)
```

local update is implemented through minibatch SGD algorithm. It is worth noting that due to the push only nature of the considered protocol, the merge-update-push cycles are not synchronized among participants: a node may merge its fresher model with an outdated one. The GL strategy, in [43], is not evaluated on DL architectures. Furthermore, this seminal work does not thoroughly discuss some aspects related to different kinds of heterogeneity that arise in real-world cross-device setting; in particular, the data held by peers, the neighbors reachable by each peer in the network, and the processing and communication speeds of devices are unrealistically supposed to be homogeneous. Such aspects are considered and discussed in [35], where it is claimed that gossip learning shows poor performance on restricted communication topologies and it is highlighted that GL fails to converge when communication speeds of the nodes and heterogeneity of data are correlated. Authors of [35] propose some strategies to improve GL in such realistic scenarios.

In BACombo [50], authors consider a fixed topology of neighbors for each learner, not limiting the spreading of the updates to one peer per round, and propose a neural-network specific solution. The local model held by each peer is splitted into a set of S not-overlapped segments, and each participant does not pull all the segments (i.e., the entire model) from the same peer but collects S segment from S different links in the network of neighbours. In this way, each peer reconstructs a model update by building a mixed model composed by such S segments that have been pulled from different peers. They extend the solution by allowing each peer to pull $S \times R$ segments in each round of communication, with R being an hyper-parameter, to be carefully tuned, that represents the number of mixed models that can be reconstructed, thus impacting the communication efficiency while accelerating the propagation of fresh model. The mixing strategy is similar to FedAvg, weighing contributions (i.e., segments) according to the cardinality of the dataset held by participants.

In [108], authors propose a consensus-based FedAvg-inspired algorithm (referred as CFA), supposing sparse connectivity. The algorithm is formalized in Algorithm 2. In each round, the participant k receives models from its neighbors and produces an aggregated model, ψ^k. Next, local iterations of mini-batch SGD are performed to produce the new-generation model, that will be sent to the neighbors, before the process repeats. The peculiarity of the algorithm stands in how the aggregated model is obtained, at round t, from the neighbor contributions, that is: $\psi_t^k = w_t^k + \zeta_t \sum_{i \in N_k^-} \alpha_{k,i}(w_t^i - w_t^k)$, where ζ_t is the "consensus step size" and the mixing weights $\alpha_{k,i}$ are chosen, similarly to FedAvg, as $\alpha_{k,i} = \frac{n_i}{\sum_{i \in N_k^-} n_i}$ with n_i being the cardinality of data samples at participant i.

We conclude this overview about instances of baseline algorithms for server-less federated learning by mentioning the fact that blockchain-based implementations of peer-to-peer learning frameworks have been — and are — explored in literature (e.g., [58]), though not being explored in this survey.

4 DECENTRALIZED LEARNING SOLUTIONS: A SYSTEM-LEVEL ANALYSIS

Decentralized learning decouples by design the ability to learn a predictive ML/DL model from the direct access to raw data and meets the rising urge of ensuring privacy guarantees to the data owners while still being able to distill useful information for the community. However, as already pointed out in this survey, diverse challenges emerge. Chief among them, privacy is not completely secured by means of just disclosing ephemeral updates (e.g., gradients, model parameters) or meta-level information, as well as the communication efficiency is of paramount importance in cross-device federated settings. Furthermore, having the raw data (massively) distributed and/or unbalanced among participants naturally implies dealing with non-IIDness. An additional factor to be addressed is the heterogeneity of devices' resources in cross-device settings. Moreover, the design of decentralized learning approaches opens up to new possibilities for attackers, since learners actively participate in the training process, e.g. forcing information leakage from other participants or trying to influence the behaviour of the system. These are the most investigated issues in literature so far, but other less crucial aspects and challenges are rising and taking the scene while effective solutions for the urgent aspects permit to already apply decentralized learning in real scenarios. In this section, we discuss the systems in the literature that aim at solving the above mentioned issues, i.e. communication efficiency, privacy, non-IIDness, device heterogeneity, and poisoning defense, classifying them by our taxonomy (see Table 1).

Let us note that, in the following sub-sections, we will use the taxonomy definitions and terms introduced previously in this survey; where not possible or convenient, we explain in-line the specific meaning of the employed definitions/terms/symbols.

4.1 Improving Communication Efficiency

The communication efficiency in decentralized learning can be addressed from different perspectives. In the first place, decentralized optimization algorithms are usually designed to allow for multiple local training iteration between communication rounds to reduce the total communication cost of the training process (e.g.,[81], [54]); in synchronous star-shaped federated learning the number of participants selected per round is typically limited (e.g., [81]), as well as in peer-to-peer topology the number of neighbours to scatter the updates to is bounded (e.g. bounded to 1 such as in GL [43] or in [117]). Stream compression (e.g., by encoding, quantization and/or sparsification of updates) is typically employed to reduce the per-round communication cost [61] [16] [103] [106] [118] [85] [67] [51] [117]. Furthermore, specific strategies can be crafted accordingly to the peculiarities of the model to train (e.g., by introducing asynchrony between the updating of the neural-network parameters belonging to shallower/deeper layers [20]). Stream compression has been mostly explored in star-shaped federated learning, but similar solutions may be easily adapted in peer-to-peer topology. An orthogonal approach is to improve the communication efficiency by reducing the total communication rounds needed for the model convergence (e.g., implementing distributed variants of SGD optimizers [85] [77] [108]). Or again, communication-efficiency can be

Table 1. This tabular classification is used to guide the readers; the referred works are characterized by the federated setting they refer to, by our taxonomy from Section 3.2, and by the most relevant issues addressed, i.e., communication efficiency (CE), privacy (P), non-IIDness (non-IID), device heterogeneity (DH), poisoning defense (PD). We flatten the update mode ramification of the taxonomy, related to data-parallel approaches, for better visualization.

Notation: w (full) model parameters, w_d on-device layer-partitioned model parameters (e.g., in SL), g gradients, lv logit vectors, A activations (i.e., output of NN's *cut layer*), Y labels associated with data points, m 1^{st} moments, v 2^{nd} Adam moments, c control variates, d GD momentum, t time stamps, res_info resource information, L loss function value, ρ the Lipschitz parameter of the loss function, β the smoothness parameter of the loss function, τ^* the optimal number of local updates between synchronizations.

* indicates that the work is not thoroughly discussed throughout the section.

				Our Taxonomy Characterization									
				On-dev.	Data		Update		Topology		Exch. Info		MEC
	Work	Year	Setting	Model	S	P	Async	Sync	Star	P2P	Up	Down	aware
Baseline	FedAvg [81]	2016	both	Full		✓		✓	✓		w	w	×
	FD [49]	2018	device	Full		✓		✓	✓		lv	lv	×
	CFA [108]	2019	device	Full		✓	✓			✓	w		×
	GL [43]	2019	device	Full		✓	✓			✓	w		×
	BrainTorrent [104]	2019	silo	Full	✓		-	-	✓		w		×
	SplitNN [36]	2018	silo	Split	✓		-	-	✓		A, Y, w_d	g, w_d	×
	SFL [119]	2020	device	Split		✓		✓	✓		A, Y, w_d	g, w_d	×
Comm. Efficiency	Kamp et al. [54]	2018	device	Full		✓		✓	✓		w	w	×
	Konečný et al. [61]	2016	device	Full		✓		✓	✓		w	w	×
	Caldas et al. [16]	2018	device	Full		✓		✓	✓		w	w	×
	STC [106]	2019	device	Full		✓		✓	✓		w	w	×
	eSGD [118]	2018	device	Full		✓		✓	✓		g	g	✓
	HierFAVG [75]	2019	device	Full		✓		✓	✓		w	w	✓
	Chen et al.* [20]	2019	device	Full		✓		✓	✓		w	w	×
	CE-FedAvg [85]	2019	device	Full		✓		✓	✓		w, m, v	w, m, v	×
	CFA-GE [85]	2019	device	Full		✓	✓			✓	w, g		×
	SAPS-PSGD [117]	2020	silo	Full		✓		✓		✓	w		×
	Momentum FL [77]	2020	device	Full		✓		✓	✓		w, d	w, d	×
Privacy	Geyer et al. [34]	2017	device	Full		✓		✓	✓		w	w	×
	DP-FedAvg [82]	2017	device	Full		✓		✓	✓		w	w	×
	Triastcyn et al. [120]	2019	device	Full		✓		✓	✓		w	w	×
	SECAGG [13]	2017	both	Full		✓		✓	✓		w	w	×
	Turbo-Agg [112]	2020	device	Full		✓		✓	✓		w	w	×
	Hao et al. [37]	2019	device	Full		✓		✓	✓		g	g	×
	SecGD* [40]	2019	silo	Full		✓		✓	✓		g	w	×
	Truex et al. [122]	2019	both	Full		✓		✓	✓		w	w	×
	SecProbe [142]	2019	silo	Full		✓		✓	✓		w	w	×
	MCL* [32]	2019	silo	Full		✓		✓	✓		w	w	×
	NoPeekNN [123]	2019	silo	Split	✓		-	-	✓		A, Y, w_d	g, w_d	×
	Yu et al. [132]	2019	silo	Split	✓		-	-	✓		A, Y, w_d	g, w_d	×
P & CE	DiffSketch* [67]	2019	device	Full		✓		✓	✓		g	g	×
	Jin et al. [51]	2020	device	Full		✓		✓	✓		g	g	×
	cpSGD* [2]	2018	device	Full		✓		✓	✓		g	w	×
	Bonawitz et al. [14]	2019	device	Full		✓		✓	✓		w	w	×
Non-IID	Y. Zhao et al. [145]	2018	silo	Full		✓		✓	✓		w	w	×
	FedAug [49]	2018	silo	Full		✓		✓	✓		w	w	×
	FedMeta, UGA [131]	2019	device	Full		✓		✓	✓		g	w	×
	FedAvgM* [47]	2019	device	Full		✓		✓	✓		w	w	×
	FedProx [69]	2019	device	Full		✓		✓	✓		w	w	×
	SCAFFOLD [56]	2019	device	Full		✓		✓	✓		w, c	w, c	×
	FedDANE [70]	2020	device	Full		✓		✓	✓		w, g	w, g	×
	FedOpt [101]	2020	device	Full		✓		✓	✓		w	w	×
	FAVOR* [124]	2020	device	Full		✓		✓	✓		w	w	×
DH	FedAsync [127]	2019	device	Full		✓	✓		✓		w, t	w	×
	TiFL [17]	2020	device	Full		✓		✓	✓		w	w	×
	FedCS [91]	2019	device	Full		✓		✓	✓		w, res_info	w	✓
	LoAdaBoost* [48]	2018	silo	Full		✓		✓	✓		w, L	w, L	×
	Wang et al. [125]	2019	device	Full		✓		✓	✓		w, g, ρ, β, L, res_info	w, τ^*	×
	BACombo [50]	2020	device	Full		✓	✓			✓	w		×

Table 1. Continuation

				Our Taxonomy Characterization									
				On-dev.	Data		Update		Topology		Exch. Info		MEC
	Work	Year	Setting	Model	S	P	Async	Sync	Star	P2P	Up	Down	aware
PD	SLSGD [128]	2019	device	Full		✓		✓	✓		w	w	×
PD	FoolsGold [33]	2018	device	Full		✓		✓	✓		g	w	×
PD	L. Zhao et al. [141]	2019	device	Full		✓		✓	✓		w	w	×
PD	Li et al. [66]	2019	device	Full		✓		✓	✓		w	w	×

architecturally favoured by leveraging MEC [75]. Obviously, combinations of the previous strategies are common.

FedAvg can be seen as a periodic averaging protocol that involves in each round of communication only a random subset of the participants. However, FedAvg (and periodic averaging protocol in general) maintains the same frequency of communication independently from the utility of the specific synchronization, e.g., when all models are approximately equal or they have already converged to an optimum then synchronization may be omitted. Leveraging this observation, authors of [54] propose a dynamic averaging protocol to invest the communication efficiently by avoiding to synchronize models when the impact of such aggregation on the resulting model is negligible. To this end, authors leverage a simple measure, $\|w_t^i - r\|^2$, for model divergence to quantify the effect of synchronizations; specifically, they measure the divergence of the locally trained model, w_t^i, for the round t at participant i, with respect to a reference model r that is common among all participants, e.g. the last received global model, and compare such divergence with an a-priori chosen threshold to decide whether perform a synchronization.

In [61], two strategies have been proposed to reduce the uplink cost in star-shaped FL (explicitly considering FedAvg as baseline) by means of compression, and they are *structured updates* and *sketched updates*. Such strategies can be combined to further compress the data to be sent from clients to server. The peculiarity of structured updates is that the updates are restricted to have a pre-defined structure, and they are directly trained to fit such structure. Two types of structures are considered by authors: (i) updates are enforced to be a low-rank matrix of rank k, with k being a fixed parameter (low-rank updates); (ii) updates are restricted to be a sparse matrix following a pre-defined random sparsity pattern (i.e., a random mask), thus only the non-zero values along with the seed to generate the pattern have to be communicated. Regarding sketched updates, the full (or structured) update resulting from the local training is approximated, i.e. sketched, in a lossy compressed form. To this end, two (compatible and jointly usable) tools are proposed: subsampling, i.e only a random subset of the (scaled) values of the updates are communicated, and probabilistic quantization. As the reader can note in the continuation, several successive works addressing communication efficiency in decentralized training combine subsampling or sparsification and quantization. Furthermore, supported by empirical evidence, authors highlight the usefulness of applying structured random rotations before quantizing to reduce the quantization error.

Similarly to [61], authors of [16] use a combination of basis transform, subsampling and probabilistic quantization to reduce the server-to-client communication cost[6] of FedAvg. Furthermore, inspired by the well-known dropout technique [114], clients train their updates considering a smaller sub-model with respect to the global model. This further reduces the server-to-client traffic, reduces the local computational cost and, obviously, reduces the client-to-server traffic. Differently from the traditional dropout, a fixed number of activations are zeroed out at each fully-connected

[6]Note that in the work [61] the objective is to reduce the client-to-server communication cost.

layer, thus all the possible sub-models have the same reduced architecture, while a fixed percentage of filters are zeroed out for convolutional layers. Authors call this strategy Federated Dropout. The client-to-server communication cost can be ultimately reduced by combining the solution of [61] and Federated Dropout. To summarize, the process works as follow: at the beginning of each round, the selected clients receive a compressed sub-model from the server; they decompress it, locally compute an update, and compress such update to send it back to the server; the server decompresses the received sub-models updates and maps them to the global (full) model either by exchanging a random seed or via state on server-side. In the end, the hyperparameters to be tuned are (i) the type of basis transform, (ii) the fraction of weights that are not zeroed out during the subsampling, (iii) the number of quantization bits, (iv) the federated dropout rate, i.e. the percentage of neurons remaining active; (i), (ii), (iii) can be different for the uplink and the downlink.

Building on their previous Sparse Binary Compression (SBC) [107] technique that targets the traditional distributed setting, in [106] authors specifically design a compression framework for cross-device federated settings. The proposed Sparse Ternary Compression (STC) compresses both the upstream and the downstream communication with respect to the baseline FedAvg while improving the robustness to non-IID data as well as to partial client participation. In addition to experimentally confirming the already known weakness of vanilla FedAvg in presence of heterogeneous data, authors also show poor model accuracy with aggressive quantization schemes, such as SignSGD[7] [9], in non-IID scenarios. Conversely, $top_{p\%}$ sparsification, i.e. dropping all but the p fraction of updates with the highest magnitude, suffers least from heterogeneous data. This observation leads the design of the proposed compression scheme for the upstream communication in FL. As happens in SBC, STC exploits (i) $top_{p\%}$ sparsification of weight deltas (i.e., the difference between the global model and the local model), (ii) local residual accumulation[8], (iii) binary quantization of the $top_{p\%}$ elements[9] and (iv) encoding (to losslessly compress the distance between the non-zero elements of the sparse weight-update) to reduce the amount of data to be sent from participants to the server. It is worth to highlight once more that this strategy alone does not affect the downstream communication. In this regard, authors observe that, although clients-to-server updates are sparse, the server-to-clients update essentially becomes dense as the participation rate, i.e the fraction of participants involved in each round, exceeds the inverse sparsity, i.e. the inverse of the hyperparameter that rules the sparsification. In fact, in the worst case, the number of non-zero elements in the aggregate (the sum) of clients-to-server updates grows linearly with the number of participating clients. The dense nature of server-to-clients updates prevent an effective compression. Therefore, they propose to apply their STC algorithm also to the aggregated updates at server side, hence the server maintains a residual as well. However, the partial client participation in each round of FL prevents a straightforward application of STC at server-side: STC sparsifies and compresses weight deltas, and, considering that not all the participants are involved in every round, some participants could not recover the updated weights from the received (compressed) delta, since they may not have participated to the previous round(s). The solution

[7]In SignSGD [9], gradient updates are locally quantized to their binary sign from clients. The parameter server gathers such binary updates and broadcasts the belief about the sign of the true gradient. The server uses majority vote on the gathered gradient updates (See Algorithm 3 in [9]).

[8]Note that, differently from [118] (presented later on), in STC (and SBC) the residual accounts for ignored weights and not for gradients.

[9]The result of the sparse weight-update binarization is a ternary tensor containing values $-\mu$, 0, μ with μ being the mean of the $top_{p\%}$ weight-updates in absolute value. STC sets all the positive non-zeroed elements to μ and all the negative non-zeroed elements to $-\mu$. Note that in SBC the resulting sparse tensor is binary instead, and the algorithm is slightly different; they independently compute the mean of all non-zeroed positive and all non-zeroed negative weight-updates; if the positive mean is bigger than the absolute negative mean, they set all negative values to zero and all positive values to the positive mean and vice versa.

adopted is to cache the last τ updates at server-side, and to require a prior synchronization step for those outdated participants before initiating the local training. Thanks to this shrewd protocol addition, the downstream communication can be effectively reduced regardless the partial client participation.

In Edge Stochastic Gradient Descent (eSGD) [118], besides tacking advantage of edge servers to scale the collaborative training process, authors propose an algorithm to reduce the uplink communication cost when exchanging gradients in a star-shaped synchronous learning framework. The solution builds on the observation that gradients, produced by iterations of mini-batch SGD optimization, are very sparse [115]; in eSGD, participants upload only a fraction (i.e., a fixed percentage) of the gradient coordinates, only the ones that are considered important, while accumulating a residual to account for ignored coordinates[10] — merely dropping these portions of gradients, even if they are small values, can hamper the model convergence [3].

To reduce the network traffic headed to the cloud, a MEC-aware extension of FL is proposed in [75], namely Hierarchical Federated Averaging (HierFAVG). Authors exploit the hierarchical architecture of such brand-new paradigm to have middle-level aggregator entities; each τ_1 local updates, edge servers gather the updates of the participants in their proximity to produce the aggregated models of their locality; each τ_2 edge-level aggregations, the cloud updates the global model (hence each $\tau_1\tau_2$ local iterations). It is worth noting that if τ_2 is equal to 1, the HierFAVG corresponds to the traditional FedAvg, while, intuitively, with τ_2 greater than 1, HierFAVG reduces the communication cost with respect to FedAvg.

From another perspective, the communication cost of decentralized training can be reduced if less rounds are needed to reach a certain target accuracy. To this end, authors of [85] empirically demonstrate the suitability of an ADAM[59]-inspired variant of FedAvg. As well known, the ADAM optimizer leverages per-parameter learning rates, 1st moment and 2nd raw moment estimates to converge faster in traditional minibatch SGD. In the proposed CE-FedAvg, participants locally compute their update by exploiting ADAM, and they send back to the server the 1st and the 2nd moment estimates as well as the locally trained model (specifically, their deltas). Thus, beyond the global model parameters, the server also aggregates the 1st and the 2nd moment estimates, that are broadcasted at the beginning of every round to the learners. Since moment estimates have the same size of model parameters, it is straightforward to note that the communication cost per round is tripled with respect to FedAvg in absence of compression. However, authors highlight that this is compensated by the faster convergence of CE-FedAvg. Furthermore, they employ compression techniques to reduce the amount of data to be sent; sparsification, quantization and encoding are used. Authors also emphasize an additional advantage of CE-FedAvg over FedAvg: in absence of a central test/validation set of data, it is difficult to tune the learning rate for FedAvg, while the default ADAM's hyperparameters seem to be suitable for general use.

Similarly, the authors of [77] implement a federated version of momentum gradient descent, namely Momentum FL, where momentum terms and model updates are exchanged between participants and server, round by round, doubling the communication cost of each round with respect to FedAvg, while taking advantage of faster convergence rate.

The same purpose, i.e. reducing the total communication rounds to reach model convergence, motivates an improvement of the CFA algorithm (already presented in 3.3.2) in peer-to-peer topology of learners. Authors propose to introduce a "negotiation" phase where, before using the aggregated model ψ_t^k to run local training, the participant k feeds back ψ_t^k to the same neighbors.

[10] Gradient sparsification and local gradient accumulation is a well-known technique in the traditional distributed setting to reduce the communication cost by speeding up the training process (i.e. less communication rounds) without significantly degrading the resulting model accuracy [115][3][74]. Error accumulation, in this case weight accumulation, permits to not waste gradient information, although they may suffer from staleness.

Neighbors compute gradients with respect to ψ_t^k, and send them back to the participant that has forwarded the request. Next, gradients are aggregated, leveraging a tunable mixing parameter, to produce $\widetilde{\psi_t^k}$ that is then used as starting point for the local learning iteration. This strategy should make the learning faster[11]. However, this algorithm requires four communication rounds, and moreover the negotiation is synchronous. Therefore, the algorithm is transformed into a two-stage algorithm, referred as Consensus FedAvg Gradient Exchange (CFA-GE) [108]: the negotiation phase is performed without the need of sending ψ_t^k and receiving back the neighbors' gradients, permitting to save communications and avoid the synchronization intermediate step (i.e., waiting for the neighbors to send back the gradients with respect to ψ_t^k). The insight is to exploit past (and outdated) models received from a certain neighbor during the previous rounds to produce, in advance, a gradient prediction for that neighbor, and this is done for all the neighbors. In this way, it is possible to scatter such gradients prediction together with the next-generation model parameters; each participant hence receives such information, produces ψ_t^k by aggregating the neighbors' model as we have seen for the baseline CFA algorithm, and uses the received gradient predictions to adjust the model to obtain $\widetilde{\psi_t^k}$, and finally applies the local training to $\widetilde{\psi_t^k}$ that will generate the updated model.

In [117], the authors propose an efficient peer-to-peer framework for cross-silo communication, namely SAPS-PSGD, where aggressive model sparsification is coupled with single-peer communication scheme. They leverage a coordinator entity – not a parameter server – that, in extreme synthesis, broadcasts to the participants a gossip matrix and other some necessary information (i.e., the current global step, a random seed to generate the mask for applying the desired sparsification) and synchronizes the rounds of communication among such node pairs. The gossip matrix is built by taking into account the peers' bandwidth to favour faster links; it dynamically determines the couples of peers that will exchange highly sparse model updates during that round.

4.2 Protecting Privacy

It may be believed that sharing gradients, model updates or meta-level information (such as outputs of layers in neural-networks) in place of raw data ensures privacy protection. However, it has been demonstrated that gradients exchanged during the distributed training process do leak information about the training data [148] [140] [40] [97] [89] [45] as well as model updates [84] [89] — even though it may be preferable to exchange model weights instead of gradients under a privacy-preserving perspective [98] — and activations [25] [132].

0The literature about protecting privacy in decentralized learning comprises diverse approaches; differentially-private mechanisms [34] [82] can be employed during the distributed training process to mask updates at the cost of reduced model accuracy [7], and relaxations of traditional Differential Privacy (DP) can be leveraged to inject less noise [120], limiting the incurred performance degradation. Data-augmentation [32] and obfuscation [46] techniques can be used in visual application to prevent reconstruction of images in the training set. Multi-party secure aggregation [13] [112] and similar techniques [40] can hide the individual contributions to the aggregator, finding its main utility in star-shaped federated learning, but producing non-negligible overheads. Additively homomorphic encryption also allows the aggregator to sum updates, thus ensuring the inscrutability of single contributions [97] while not degrading model accuracy but increasing communication cost. Combinations of DP-mechanisms with secure aggregation and additively homomorphic encryption are also explored [122] [37] to balance the weaknesses of such techniques. Minimizing distance correlation between raw data and activations (at cut layer) [123] and step-wise activation functions [132] are used to prevent the invertibility from intermediary representations in the context of

[11]The negotiation phase, from an high-level perspective, can be thought to be similar to the approach of [70].

privacy-preserving Split Learning.

The first works enforcing participant-level (ϵ, δ)-DP [29] in federated settings are most notably [34] and [82]. The aim, common to both the works, is to ensure that a model trained with FedAvg does not reveal whether a certain participant has been involved during the decentralized training process, balancing the trade-off between privacy loss and model performance. It is worth highlighting that the proposed solutions protect the whole client's dataset differently from [1] where a single data point's contribution in the trained model is protected.

Authors of [34] use two randomized mechanisms to guarantee client-level DP: (i) random subsampling of participants for a certain round of communication; (ii) Gaussian mechanism. In FedAvg, the central aggregator averages the participants' updates, that here are considered to be weight deltas (i.e., the difference between the received parameter weights and the locally computed parameter weights). The key idea of [34] is to perturb and approximate such averaging (i.e. perturbing the sum of updates) by employing a Gaussian mechanism. As usual, the Gaussian-distributed noise has to be calibrated according to a certain sensitivity; such sensitivity is calculated as the median norm of all the gathered updates[12] and the updates are scaled according to such sensitivity, i.e. clipped updates. To keep track of the privacy loss within subsequent communication rounds, authors use the moments account of [1] instead of the privacy amplification lemma and the standard composition theorem [29] to obtain tighter bounds. In particular, they stop the collaborative training once the (cumulative) δ, that represents the likelihood that a participant's contribution is disclosed, becomes greater than a threshold.

The approach of [82] is slightly different from [34]. Authors, in fact, randomly sample participants by selecting each independently with probability q, hence producing variable-sized samples of participants and influencing the sensitivity of (weighted) average queries — in [34] a fixed number of clients is randomly selected. Two different bounded-sensitivity estimators are proposed to account for such participant-sampling process. Furthermore, two clipping strategies are evaluated for multi-layers models: (i) flat clipping, i.e. using an overall clipping parameter, or (ii) per-layer clipping, i.e. treating the parameters of each layer as separate vector and using per-layer clipping parameters, motivated by the observation that such vectors may have vastly different L_2 norms — anyway the clipping parameter is fixed throughout the training process, while in [34] is dynamically calculated as the median norm of all the unclipped contributions.

In [120], authors allocate a tighter privacy budget for guaranteeing client-level DP and instance-level DP, i.e. less noise to reach the same privacy guarantee, also improving the accuracy of the trained model. They employ a relaxation of traditional DP, in this case Bayesian DP (BDP) [121], by making two assumptions (i) stationary data distribution and (ii) datasets with unchangeable samples. Authors also use a Bayesian accounting method instead of state-of-the-art moments accountant [1] thanks to the assumption that data come from a particular distribution and not all the data are equally likely; this observation can lead to sharper privacy loss bounds with BDP in federated setting. Besides the proposed use of BDP, to limit the noise added to guarantee both instance-level and client-level DP, the noise to be added by the server for client-level DP is "re-counted" considering the injected noise during the on-device gradient descent. They call this approach joint accounting. However, a limitation emerges: joint accounting is only usable for FedSGD algorithm, not for FedAvg (because the possible multiple local iterations in FedAvg, hence multiple noisy steps, may influence the point at which the gradient is computed: a different gradient distribution can arise or the total noise variance can be underestimated).

[12]The sensitivity is calculated by the server in each communication round.

To prevent the server from peeking in individual updates during the aggregation phase, a practical protocol for secure aggregation, namely SECAGG, has been proposed in [13] for federated settings — reminding that the communication bottleneck and the dropping of users are peculiar of such scenarios. In a nutshell, star-shaped FL systems leverage a central server that computes sums of updates from which deriving the new-generation global model round by round. The scope of SECAGG is to hide the individual contributions of participants and release only the sum of such updates to the server, preventing privacy violations from the aggregator entity. The essence of the approach is similar to differential privacy: updates are locally perturbed, but, while in DP-mechanisms such perturbations become part of the updates (they are never removed, in fact noise calibration is fundamental to not compromise the training), in SECAGG such perturbations are neutralized during the aggregation phase. The insight is to have pairs of participants — hereinafter referred as participant u and participant v — that share randomly sampled 0-sum pairs of mask vectors, $p_{u,v}$ and $p_{v,u}$; before uploading their model updates, participants u and v add such masks to their contributions, with $p_{u,v} + p_{v,u} = 0\ \forall u \neq v$; each participant u computes a random mask vector and perturbs (i.e., adding $p_{u,v}$ if $u > v$ or subtracting $p_{u,v}$ otherwise) its local update for each other user v; mask-pairs are canceled out during the sum of all contributions. Every pair of participants share a common random seed $s_{u,v}$ of some fixed length that can be fed to a secure Pseudorandom Generator PRG [11] to generate the mask pairs, hence the seed can be transmitted in place of the the entire mask (that has the same size of updates) reducing the communication burden. These shared seeds are established through Diffie-Hellman [23] key exchange, composed with a hash function. It is worth noting, that (i) SECAGG requires the elements of the input vectors, i.e. the participant's updates, to be integers $modK$, while (ii) the elements of the vector updates are typically real-valued instead, and that (iii) the employed PRG's output space is the same of the input space. Therefore, the real-valued elements of the updates are typically clipped to a fixed range of real numbers, and then quantized among such range using k bins, and the SECAGG modulus is chosen to be $K = kn$, with n being the number of participants.

A practical protocol for collaborative training in federated settings must be able to tolerate a fraction of dropping users. To this end, SECAGG leverages Shamir's t-of-n Secret Sharing [109] to permit recovering the pair-wise seeds of a limited numbers of dropping participants; in practice, each participant sends encrypted shares of its Diffie-Hellman secret to all other participants via server. SECAGG also accounts for the critical case in which a certain participant belatedly responds to the server with its contribution by using a double masking for the updates. In addition to $p_{u,v}$, a private mask vector p_u (generated from a seed b_u as well) is further added to the update, and also its shares are distributed during the secret sharing round for the pair-wise masks.

SECAGG has been employed in the FL system designed in [12] but highlighting that the quadratically grow (with respect to the number of participants) of the computational cost for the server limits the maximum size of an instance of SECAGG to hundreds of learners. They indeed leverage intermediate secure aggregators for subsets of participants, and the intermediate sums are further aggregated without SECAGG by a master aggregator.

A recent work [112], namely Turbo-Aggregate, addresses the quadratic growth of the computational cost and of the communication overhead by slightly changing the approach, and still being resilient to user dropouts (up to 50% of participants). The key idea is to partition the federation of learners in groups that actively participate in the aggregation and dropout-recovery phases instead of just leveraging the central server, and to add redundancy directly in the model updates to reconstruct the missing contributions of dropout participants instead of Shamir's t-of-n Secret Sharing such as in SECAGG. In a nutshell, reminding that the scope is to securely compute a sum (i.e., the sum of locally computed updates) and assuming that all communications take place via central server employing Diffie-Hellman key exchange protocol, Turbo-Agg works as follow.

Firstly, participants are randomly divided in L groups, with each group being composed of N_l participants. The set of participants in group l is referred as U_l. The process involves L stages, and Turbo-Agg adopts a circular and sequential strategy in its simplest version: in each stage only one group is involved; the output produced from a group in a certain stage is the input for the next group[13]. Ignoring for a moment the possibility of dropout, in each stage, the participant i in group l masks its update $x_i^{(l)}$ with a random vector $u_i^{(l)}$ being known (and communicated) only by the honest server, similarly to what happens in SECAGG. To be secure against server-participants collusion, learner i additionally masks its update with another random vector $r_{i,j}^{(l)}$, and the resulting masked update $\tilde{x}_{i,j}^{(l)} = x_i^{(l)} + u_i^{(l)} + r_{i,j}^{(l)}$ is sent to each participant j of the group $l+1$, with $\sum_{j\in[N_{l+1}]} r_{i,j}^{(l)} = 0$, i.e. random vectors r cancel out during aggregation. The secure sum is cooperatively computed, group by group, and can be summarized thanks to the recursive relation $\tilde{s}_i^{(l)} = \frac{1}{N_{l-1}} \sum_{j\in[N_{l-1}]} \tilde{s}_j^{(l-1)} + \sum_{j\in[U_{l-1}]} \tilde{x}_{j,i}^{(l-1)}$ with $\tilde{s}_i^{(l)}$ that is a variable locally held by each participant i in group $l > 1$, and that represents the aggregated masked updates from the previous group[14]. It is important to highlight that each participant i of group l sends $\tilde{s}_i^{(l)}$ and $\tilde{x}_{i,j}^{(l)}$ to each learner j of the group $l+1$. A final aggregation step is necessary to preserve the privacy of the participants in group L at the stage L; an additional group (referred as $final$), in fact, is randomly composed (for example, among the survived learners) with each participant aggregating the contributions coming from the group L, and sending the results to the server. Specifically, participants j in the $final$ group produces $\tilde{s}_j^{(final)} = \frac{1}{N_L} \sum_{i\in[N_L]} \tilde{s}_i^{(L)} + \sum_{i\in[U_L]} \tilde{x}_{i,j}^{(L)}$ and send it to the server, that can recover the sum of unperturbed updates by applying $\frac{1}{N_{final}} \sum_{j\in[N_{final}]} \tilde{s}_j^{(final)} - \sum_{m\in[L]} \sum_{j\in[U_m]} u_j^{(m)}$. However, in case of participant dropouts the protocol will fail, since, for example, the random vectors r cannot be cancelled out. To this end, authors propose to employ Lagrange coding [134] to allow participants of group l to recover the missing contributions from group $l-1$, and to compute the partial aggregation anyway. Being concrete and redirecting to the full paper [112] and to [134] for theoretical detail, each participant has to send to each participant j in group $l+1$ two additional (coded) vectors in each stage, namely $\bar{s}_i^{(l)}$ and $\bar{x}_{i,j}^{(l)}$, in addition to $\tilde{s}_i^{(l)}$ and $\tilde{x}_{i,j}^{(l)}$. The employed coding strategy allow each learner in group $l+1$ to reconstruct the vector $\{\tilde{s}_i^{(l)}\}_{i\in N_l}$ starting from at least N_l evaluations (i.e, $\bar{s}_i^{(l)}$ and $\bar{x}_{i,j}^{(l)}$) from the previous stage. Therefore, since each participant send two evaluations to the learners in the next group, this redundancy permits to tolerate up to half of learners dropping.

It is worth noting that, although SECAGG and its variant Turbo-Aggregate explicitly targets star-shaped networks of learners, they are suitable for fully decentralized networks, i.e. peer-to-peer topologies, with one peer (or more) working as aggregator.

An alternative to SECAGG for star-shaped FL frameworks is represented by Additively Homomorphic Encryption; since such technique guarantees the additivity of multiple ciphertexts, the server can perform the aggregation without the need of seeing the updates in clear. In [37], authors propose to use a symmetric additively homomorphic encryption called PPDM [146] for its efficiency, combining it with Laplacian mechanism for DP in order to neutralize collusion between compromised users and malicious server. They show drastically reduced communication overhead with similar solution [97], that employs paillier encryption instead.

In [122], authors combine multi-party computation (MPC) via Threshold Homomorphic Encryption and Differential Privacy to balance their respective weaknesses; in fact, applying DP to provide

[13]Since only one group is active per stage, for ease of notation, group and stage are referred both with the index l.

[14]The initial aggregation at group $l = 1$ is set as $\tilde{s}_i^{(1)} = 1$.

the required level of privacy may degrade accuracy while MPC alone is vulnerable to inference attacks over the output, i.e. the intermediate models during the collaborative training process and the final predictive model. Leveraging only on one of those two techniques may compromise the effectiveness of the system (in terms of prediction accuracy of the resulting model or in terms of privacy guarantee). The key intuition in [122] is to reduce the traditional amount of locally-injected noise to ensure ϵ-DP by exploiting the MPC framework building on the assumption that t participants are trusted (i.e., non-colluding parties), with t being a customizable parameter; thanks to this assumption, the Gaussian noise to be added to each local query is reduced by a factor of $t-1$. In the worst scenario, the performance (in terms of model accuracy) of the proposed system converges with existing local DP approaches.

00Considering the scenario in which the data quality of certain participants, namely *unreliable participants*, may be poor (meaning that a portion of their data is not always accurate as the data held by others), authors of [142] focus on guaranteeing two levels of privacy: (i) preserving privacy of the participant's data and (ii) hiding the eventual participation in the training process of unreliable participants. At the same time, they focus on limiting the impact on the global model of such participants. The proposed solution, SecProbe [142], ensures participants' privacy by perturbing, during the local training process, the objective function of the neural network using the functional mechanism (FM) [138] to achieve ϵ-DP, and obtaining the sanitized parameters by minimizing the perturbed objective function.

To make the metadata exchanged in Split Learning irreversible, in [132] authors propose to modify the conventional activation functions to be step-wise, i.e. the activation function is discretized by having the input domain divided into intervals and the output constant for each interval; in this way, it is not possible to exactly recover the activations' input from their outputs[15]. In this context, another approach to reduce invertibility of intermediate representations consists in minimizing the distance correlation between raw data and the communication payload, i.e. having a low distance correlation while maintaining the accuracy in predicting the output labels. Authors of [123] hence train the neural network by using a weighted combination of two losses as loss function, and such losses are the log distance correlation [116] and the categorical cross entropy. The former is used as a measure of statistical dependence between the input data and the estimated cut layer activations, while the latter traditionally considers the true labels for the inputs and the predicted labels. Intuitively, the distance correlation is minimized to ensure privacy and the cross entropy is minimized for classification accuracy. The solution is evaluated on visual datasets.

4.3 Combining Privacy and Communication Efficiency

Lossy compression techniques inherently lead to a privacy improvement, however it is not straightforward to measure the effective privacy guarantees, for example under DP formalism. The works surveyed in 4.1 do not explicitly measure privacy, and the ones in 4.2 do not address the communication cost as primary concern, while examples of combined approaches can be found in [67] and in [51]. Furthermore, other aspects in conjugating privacy and communication efficiency emerge; the secure aggregation protocol [13] can be redesigned to account from the beginning for communication efficiency [14], while tailored DP-mechanisms can be more amenable to privacy analysis when quantization of noisy DP-updates is employed[2].

[15] Authors of [132] consider three activation functions: sigmoid, hyperbolic tangent and ReLU [87]. While sigmoid and hyperbolic tangent are bijective functions, ReLU is a surjective function, and the output of ReLU can be reversed only if the input is positive. The proposed solution "masks" the output of such positive inputs by using a step-wise variant of ReLU.

In [51], authors combine communication efficiency, privacy guarantees and resilience to malicious participants under non-IID data distribution. They consider a star-shaped synchronous collaborative learning framework in which participants and server exchange (aggressively compressed) gradients instead of model parameters. The proposed algorithms use as baseline the SignSGD [9] algorithm with majority vote, that, however, does not explicitly and formally address privacy protection of participants and that has been shown to fail to converge when the data on different learners are heterogeneous [19] [106]. In particular, to deal with non-IID data, authors first propose a variation of SignSGD, namely *sto-sign*, that applies a two-level stochastic quantization on locally computed gradients, and then only transmits the signs of such quantized values. Additionally, *dp-sign*, a differentially private version of *sto-sign*, is designed to ensure formal privacy guarantees for participants involved in the training. Authors theoretically relate the Byzantine[16] resilience, i.e. the number of Byzantine workers that can be tolerated without harming the convergence guarantees, of their proposed algorithms to the heterogeneity of local datasets. Authors also propose an extension of their algorithms which takes account for residual error on server side and uses it to correct the majority vote. The convergence of the proposed algorithms is established theoretically.

With respect to just sending the quantized updates in clear, the SECAGG[13] protocol leads to a bandwidth expansion[17] that is less than 2x while ensuring reliability of the secure aggregation to dropping or collusion of a fraction of users. However, in [14], authors critically observe some limitations of a straightforward combination of SECAGG and compression techniques; chief among them (i) quantizing to a fixed point representation requires selecting the clipping range $[-c, c]$ a priori that may be challenging to establish or may lead to poor approximations if the clipping range is not large enough, and (ii) the SECAGG modulus is chosen to be $K = nk$ to represent all possible aggregated vectors without overflow (for example, if clients are 2^{10} the SECAGG modulus are 10 bits wider than they would be without accounting for secure aggregation) dominating the communication cost introduced by SECAGG — the bandwidth expansion determined by secret sharing and cryptography is much less influential. The scope of [14] is to propose a recipe for an auto-tuning (observation (i)) communication-efficient (observation (ii)) secure aggregation. The key idea is to avoid clipping at client-side but instead quantizing over an unbounded range according to a quantization bin size b that is dynamically and tightly adjusted by the server (and communicated round by round) according to the distribution of the entries of the sum relative to the previous round, and then locally applying the *mod k* operation instead of clipping; the server can compute a tight bin size b exploiting the assumption that the entries of the sum fit a normal distribution thanks to a random rotation that is locally performed by the participants (before quantizing) to their updates.

4.4 Addressing non-IIDness

As empirically shown by [81], carefully tuning the number of local epochs is crucial in FedAvg since during additional on-device iterations — less frequent synchronization among participants — local models can significantly drift apart from the global model potentially preventing convergence. Such an issue is exacerbated when considering statistically heterogeneous data from different participants [81] [145] [107] [47] — realistic assumption especially in cross-device federated settings. Data sharing and data augmentation techniques have been demonstrated to effectively alleviate the impact of non-IIDness at the cost of less decentralization [145] [131] [49]. Another major line of works tackles the problem by directly limiting the drift of the model's objective function by

[16] A Byzantine participant may transmit arbitrary information. Authors of [51] assume that such Byzantine participants upload the opposite signs (the opposite sign of each entry) of the true gradients, with the true gradients being the average gradients of all the normal workers (hence, it is supposed that the attackers know such quantities).

[17] 1.73x bandwidth expansion considering 2^{10} participants (i.e., $n = 2^{10}$) and 16 bit fixed point representation (i.e., $k = 2^{16}$).

means of proximal terms or/and gradient correction terms at the (possible) cost of communication overhead [69] [56] [70] [127]. Or again, employing SGD optimizers, such as server-side momentum [47], and, more in general, adaptive gradient-based optimizers [101], i.e., incorporating adaptive learning rates, have been shown to mitigate the effect of heterogeneous data as well as reducing the total communication rounds to reach model convergence. Also experience-driven solutions have lately emerged to counterbalance non-IIDness and speed-up convergence; a deep reinforcement learning based mechanism that intelligently selects the participants for each FL round has been proposed in [124].

Authors of [145] experimentally show that test accuracy of FedAvg can be significantly increased in non-IID scenarios by providing a small subset of globally shared data (e.g., 5%); participants use their private dataset augmented with such data examples, provided by the server, to train their updates. Despite the effectiveness of the proposed solution, it has the cost of less decentralization and requires communicating the globally shared data to the participants. Authors also propose an alternative initialization of the global model; instead of a random initialization, the server trains a warm-up model using the shared data before broadcasting the model at the beginning of the learning task.

Authors of [131] observe two critical aspects of FedAvg, especially when dealing with non-IIDness. In fact, they argue that the additional on-device iterations between global synchronizations produce gradient biases, and that selecting a fraction of participants in each round results in an inconsistency between the optimization objectives and the real target distribution (the global model is trained by minimizing the empirical loss on data distributions that are, in general, different in each round of FedAvg). Since allowing multiple local iterations and selecting a part of clients are fundamental for the communication efficiency of FedAvg and its suitability in federated settings, authors of [131] propose two (distinct but jointly usable) strategies to alleviate such issues. They propose an Unbiased Gradient Aggregation (UGA) that performs what they call *keep-trace gradient descent optimization* for the first $E-1$ epochs, and then uses the whole data set to evaluate gradients during the last epoch. The key idea of *keep-trace gradient descent optimization* is preserving the functional relation, between $w_t^{k(i)}$ and $w_t^{k(i-1)}$ in round t for subsequent on-device iterations i on client k (as usual, w indicates local/global model parameters) instead of passing for numerical values of gradients $g_t^{k(i)}$, such that in the last epoch they can calculate the gradient, g_t^k, against w_t directly (considering the entire participant's data set). It is worth noting that, in UGA, the server gathers and aggregates thus calculated gradients g_t^k to produce the global model for the next iteration. On the other hand, to address the lack of a clear objective among subsequent rounds with different participants, authors propose FedMeta. The optimization process becomes a two-stage optimization: after each global aggregation (either performed following the baseline FedAvg or UGA), the server runs an additional gradient descent step using a special dataset, $\mathcal{D}_{meta}$. The rationale is that using such meta training set at server-side provides a clear and consistent objective in the learning process. Obviously, the composition of $\mathcal{D}_{meta}$ is critical.

Authors of FedProx [69] tackle the potential model drift caused by non-IIDness by adding a proximal term to the local objective function instead of just heuristically tuning the number of local epochs; intuitively, the impact of local data is limited by restricting the locally-computed updates to be close to the current global model. Furthermore, FedProx allows for local solvers of choice, not limiting them to be SGD as happen for the traditional FedAvg. It is worth noting that FedProx is a generalization of FedAvg; if the multiplicative (hyper)parameter, μ, that rules the proximal term in FedProx is set to 0 and the local solver of participants is restricted to be SGD, FedProx exactly matches FedAvg.

Authors of SCAFFOLD [56] address the issue of drifting clients using control variates in FedAvg. The idea is to align client updates by applying a correction term to the local gradients on each local step. Each client computes its local control variate that represents the expected direction of the local update while a global control variate that represents the aggregated direction in which the server updates the global model is defined to be the uniform average of local control variates. Each participant corrects its update by adding to the locally computed stochastic gradient the difference between the global and the local control variate. The hypothetical case that motivates this strategy is to have all clients computing the same update for the global model hence eliminating the model drift. However, to achieve this, clients should communicate with each other every (either directly or via parameter server) local gradient step, e.g. each client communicating its locally computed gradient, that is unfeasible. Therefore, the local control variates and consequently the global control variates are estimated throughout the process, and the global control variate is broadcasted to the participants together with the model parameters at the beginning of every round by the server.

FedDANE [70], inspired by DANE [110] and its inexact variant [102], combines the use of the proximal term exploited in FedProx with a gradient correction term similarly to SCAFFOLD. The update phase is a two-step process: to compute the gradient correction term and to inexactly solve the Newton-type sub-problem, the locally computed gradients of the local objective functions should be firstly collected and then averaged to approximate the full gradients. However, given the realistic connection bottleneck in cross-device federated settings, it is unfeasible to gather all the locally computed gradients; in FedDANE, the full gradients are approximated aggregating the gradients of a randomly sub-sampled set of participants. It is worth noting that each update requires two rounds of communication differently from the baseline FedAvg, FedProx and SCAFFOLD — even though SCAFFOLD has to communicate in each round both the model parameters and the control variates. Despite the theoretical convergence guarantee, FedDANE shows "disappointing performance" in experimental evaluation compared to FedAvg and FedProx leaving doubts on the robustness of theoretical assumptions.

Authors of [101] propose an approach to decouple server and client learning rate and to exploit adaptive learning rates on both client and server, with the primary objective of tackling client drift. The idea is to have clients that leverage some *client optimizer* to minimize the loss on their local dataset, while the server exploits a gradient-based *server optimizer* to minimize the loss across participants. Building upon such general framework, namely FedOpt, they introduce and evaluate some per-coordinate adaptive methods as server optimizers with SGD as client optimizer. In practice, they implement three adaptive server optimizers, i.e. FedAdaGrad, FedYogi, and FedAdam respectively being the federated versions of the well-known AdaGrad [83] [27], Yogi [136], and ADAM. In their comparison with FedAvg[18] they also include FedAvgM [47]. They show that such approaches are effective, in some circumstances "dramatically" effective with respect to FedAvg, in mitigating client drift and, as a natural consequence, in reducing the total number of communication rounds required for model convergence. Authors of [101] also provide theoretical convergence analysis, and observe the need for a decaying learning rate at client-side.

4.5 Handling Device Heterogeneity

Device heterogeneity, i.e. device with diverse hardware characteristics or/and with different connectivity (in general referred as *resources*), is common in cross-device federated settings. Such heterogeneity negatively influences the training process; for example, in federated learning frameworks

[18] It is worth noting that, under the proposed framework, FedAvg and FedAvgM [47], i.e. FedAvg with server-side momentum, become specializations of the FedOpt family; the former uses SGD as both client and server optimizer with server learning rate equal to 1, while the latter employs SGD with momentum as server optimizer.

that leverage synchronous rounds, the slower participants dictate the pace if any counteraction is taken.

Authors of [127] claim that the synchronous nature of FedAvg can limit the scalability, the efficiency and the flexibility of the FL framework. In fact, (i) only few hundreds of participants are selected per round due to avoid server-side congestion (the server broadcasts the global model at the beginning of every rounds to all the selected participants); (ii) given the heterogeneity of training devices (e.g., there could be significant diversity in terms of computational power), the server usually sets a timeout for receiving back the updates and then synchronizing the model. It could happen that the selected participants that are able to complete the round within such timeout are not enough to produce a reliable update (i.e., less than the minimum participant goal count) [12]. By leveraging asynchronous updates, FedAsync avoids server-side timeouts and abandoned rounds as well as not requiring to broadcast the model to all the selected participants at the same time. Moreover, to limit the effect of staleness, a well-know drawback of asynchronous SGD approaches, FedAsync uses a weighted average to generate the new global model after aggregation as happens in SLSGD, relying a mixing hyperparameter that weighs the freshness of the aggregated model. Furthermore, to deal with drifting clients and non-IIDness, a proximal term in the local objective functions is employed as it happens in FedProx. Different alternatives are proposed to account for staleness, and to adaptatively decrease the mixing hyperparameter that rules the average in function of staleness, i.e. less weight associated with larger staleness.Under the same communication overhead, they show that FedAsync converges fester than FedAvg when staleness is small while the two approaches have similar performances considering large staleness for FedAsync. Authors state that, in general, the convergence rate of FedAsync is between single-thread SGD and FedAvg.

Asynchronous approaches, such as FedAsync [127], limit the influence of resource-constrained devices on the collaborative training process — synchronization among participants requires to wait for the slowest. In TiFL [17], authors design a system to alleviate the stragglers problem without relaxing the synchronization of FedAvg, but by clustering participants in tiers with similar response latency per round, while in LoAdaBoost [48], authors propose to use the cross-entropy loss information to early stopping the local training.

Besides asynchronism and tier of participants with similar response latency, a natural solution to address straggler clients in FL frameworks (resource constrained devices and/or devices under poor network condition) was priorly proposed in [91], in their FedCS. The goal is to maximize the number of updates to be aggregated within a specific deadline, since involving a larger fraction of participants in each round typically reduces the time needed to achieve a certain model accuracy [81]. Taking advantage of the MEC infrastructure, authors propose to extend the FL algorithm by replacing the random selection of clients with a two-step client selection; the MEC operator asks random clients to provide their resource information (computational capacities, wireless channel states, size of the dataset relevant to the current training task) from which deciding whether including them in the current training round according to an estimation of the time necessary for such participants to complete the download-train-upload process.

In [125], authors address the problem of dynamically adapting the global aggregation frequency (in real time) to optimize the learning process with a given resource[19] budget targeting a star-shaped FL framework in edge computing environments. They consider M types of resources that can be taken into account, and define that all the participants consume c_m units of type-m resource at each local update step, and each global aggregation consumes b_m units of type-m resource (with $c_m > 0$, $b_m > 0$). Being T, the number of total local update steps for the training process, and

[19] Authors of [125] consider a general definition of "resources" including, e.g., bandwidth, energy, time and monetary cost.

being τ, the number of local updates between two global synchronizations, and considering the resulting number of global synchronizations K, i.e. $K = T/\tau$, the total amount of consumed type-m resource is $(T+1)c_m + (K+1)b_m$, noting that the additional "+1" accounts for computing the last loss value after the last synchronization K. The objective is to minimize the global loss function by tuning τ and K (and, consequently, T) such that the total amount of consumed type-m resource is not greater than the resource budget R_m (each type-m resource has a certain budget associated). Such minimization problem is approximately solved by leveraging a theoretical convergence upper bound of the canonical distributed gradient descent after T iterations, although assuming that the loss function is (i) convex, (ii) ρ-Lipschitz and (iii) β-smooth. In the convergence analysis, authors also define an upper bound for gradient divergence, i.e. an upper bound of the divergence between the gradient of the local loss function and the gradient of the global loss function, that depends on how the data is distributed among different participants, hence taking into account the non-IIDness of data. We redirect to the full paper for the complete theoretical analysis. In a nutshell, the proposed control algorithm recomputes the optimal[20] τ, hereinafter referred to as τ^*, during each aggregation step via linear search on integer values of τ accordingly to the most updated parameter estimations needed to approximately solves the minimization problem mentioned above.

In regards to peer-to-peer frameworks, BACombo (already presented in 3.3.2) interestingly leverages a bandwidth-aware worker selection, i.e the peers to be requested for model segments are not trivially chosen randomly. To reduce transmission time, peers with faster network connections should be preferred. However, it is not easy to know the network condition of a certain peer a priori. The proposed solution exploit a multi-armed bandit algorithm [5]; each participant, with probability ϵ, either explores the network conditions of peers by selecting them randomly or exploits its already acquired knowledge — each participant maintains a table, that is updated each time a peer is picked for communication, that contains historical indications about the network state of that peer — by greedily selecting the peers with best network conditions.

4.6 Defending against Poisoning

From being passive data providers, in cloud-based ML, participants become active entities in the learning process of decentralized training: they locally compute updates and observe intermediate model states. Although this design is the cornerstone to improve several aspects of traditional ML/DL, it exposes the system to a larger variety of attacks from malicious learners, since participants, in theory, can contribute with arbitrary updates, and could try to manipulate the learning process for diverse scopes (e.g., merely hampering the convergence, forcing other participants to over-expose their contribution or backdooring the system), while making their detection harder since the raw data are not accessible. This is known as model poisoning, besides the more traditional data poisoning. We redirect the reader to [79] for a complete understanding of the threat model and of the attack variety. We present here some strategies to detect and/or neutralize poisoning attacks.

Authors of [128] (SLSGD) propose a variation of FedAvg to address non-IIDness and to tolerate data poisoning attacks (evaluated by simulating the attack through label flipping). They act on the baseline FedAvg algorithm by varying (i) the aggregation step and (ii) the new-model generation step; (i) instead of aggregating the updates by averaging, they use a trimmed mean to (try to) filter out poisoned updates, and (ii) instead of replacing the previous global model with the resulting aggregated model, they use a moving average between the previous and the just aggregated model

[20] It is worth noting that, intuitively, if the resource budget is unlimited, τ^* is equal to 1, i.e. global synchronization after each local update, while in presence of budget constraints it may be convenient investing the resource for local computations rarefying the global synchronizations, i.e. $\tau^* > 1$.

to limit the influence of non-IID datasets and to mitigate the extra variance caused by such "robust" aggregation.

In [33], authors propose a defense against sybil-based poisoning (precisely, label-flipping and backdoor poisoning), namely FoolsGold, targeting a federated learning framework where participants upload locally computed gradients to the (honest) aggregator. The idea is to identify malicious colluding participants, i.e. poisoning sybils, by monitoring the diversity of participants' update; sybils are supposed to share a common objective and the directions of poisoning gradients should seem unusually similar respect to updates from honest learners. In a nutshell, FoolsGold maintains an historical aggregate of updates per participant at server side, i.e. the cumulative sums of its updates so far, and it measures the cosine similarity between couple of participants' historical aggregates before each aggregation step — the rational behind this strategy is that gradients resulting from single local iteration of SGD can be very similar in directions even among honest clients, however colluding parties will share the same objective in the long run, limiting the effectiveness of poisoning throughout the training process by accordingly re-scaling the learning rate of participants that are deemed as possible sybils. The clear limit of FoolsGold — apart from being incompatible with secure aggregation and assuming honest aggregator — is that it is designed to look for sybils, hence a single participant adversary can remain undetected.

Authors of [141] propose a defense against poisoning, specifically targeting label flipping and semantic backdoor attacks, in a synchronous federated learning framework accounting also for non-IIDness. Differently from FoolsGold [33], their strategy actively leverages on clients; the server asks to the participants to evaluate some sub-models, each one derived from the aggregation of disjoint subsets of the model updates related to a certain round, and they provide back to the server an indication about the correctness in the classification task of such sub-models, tested on their private dataset, in the form of a binary matrix (obviously, a certain participant cannot receive a sub-model derived from its own contribution). Thanks to the gathered matrices, the server computes a penalizing coefficient for each sub-update to weigh the aggregation of such sub-models (for example, if more than half of the clients report the anomaly for the same sub-model, it should be zero-weighed). Authors highlight that their solution can be also combined to FoolsGold [33], e.g. to detect single-participant attack.

Similarly to [33] and [141], authors of [66] use a server-side pre-trained autoencoder model to detect abnormal weight updates that are then accordingly penalized during the aggregation.

5 OPEN PROBLEMS AND FUTURE DIRECTIONS

As an obvious observation, we remark that data-sequential approaches are only limited to Cross-silo federated settings, where the number of participants is limited (see Table 1). At the same time, (data-parallel) star-shaped synchronous systems and related improvements (i.e., 44 out of 53 surveyed solutions) have dominated the early years of decentralized learning, pushed by the Google's FedAvg baseline and, not surprisingly, the first real-world large-scale decentralized learning system for Cross-device federated settings has followed this trend [12]. Nevertheless, we stress the evidence that relaxing the synchronous constraint for aggregating updates in star-shaped systems mitigates the struggles in handling a large amount of heterogeneous devices, while introducing degrees of uncertainty that hamper the theoretical comprehension of the system's behaviour in real scenarios (e.g. FedAsync solution adopts this strategy). At the other end of the spectrum, we observe a reduced portion of fully decentralized solutions (only 5 systems out of 53, with one of them, i.e. SAPS-PSGD [117], that leverages a central entity for coordination). In addition, the MEC-architecture has demonstrated to effectively help in scaling the learning process and is increasingly adopted; in Table 1, we report 3 works explicitly considering this architecture. Indeed, that allows to favour the exploration and ease the implementation of hierarchical solutions, such as star-shaped both between devices and edge servers, and between edge servers and the cloud. To conclude, in the next subsections, we will present other open challenges that will likely influence the incoming future of decentralized learning systems, by also sketching possible and most promising directions for future research.

5.1 Rethinking the Traditional ML Workflow for Federated Learning

The literature explored in this survey proposes solutions to the main challenges of employing federated learning systems in real-world scenarios. However, most works suppose that the hyper-parameters (e.g., the neural network's architecture, regularization techniques, and optimizers) of the model to be trained have been already established, and typically the focus is not about the tuning of their determination. Furthermore, decentralized learning systems introduce additional algorithm-specific hyperparameters (e.g., the number of local epochs or the number of participants involved per round) that significantly influence the performance of the adopted solution. While in cloud-centric DL it is feasible to run many rounds of training to empirically search the hyperparameters space towards optimality, this strategy is probably infeasible for cross-silo settings and surely incompatible with cross-device settings. Hence, we expect that hyperparameter optimization that targets the communication and computation overhead on the devices that compose the federation, and not only aiming at the best accuracy of models as happens in datacenter optimizations, will gain traction, by fostering the development of easy-to-tune and/or auto-tuning algorithms for federated settings (e.g., [14] – explored in Section 4 – and [41]).

Another relevant phase of the traditional workflow in cloud-centric ML, which is reshaped by the design of decentralized learning systems, relates to the debugging of trained models' behaviour. In fact, preventing the access to the raw data by design does preclude modelers and practitioners from directly investigating the causes of the detected problems (e.g., investigating missclassification, noticing evident bias in the training set, identifying outliers, manually adding or adjusting labels), i.e. manual data inspection is impossible [6]. Connected to that, the design and implementation of privacy-preserving techniques to enable the debug phase also for federated learning systems are open areas of research. For example, in [6], the privacy concerns are overtaken by using privacy-preserving Generative Adversarial Network trained in a federated fashion, thus enabling the debug on synthetic data examples that conjugate the trade-off between information leakage and debugging utility.

5.2 Designing Incentive Mechanisms

Another assumption typically made in the FL-related literature is that the (selected) learners are willing to participate. Leaving aside for a moment the privacy concerns that may discourage participants, another factor that can determine the reluctance in being involved in federated learning processes is the associated overhead, in terms of computation and communication. Self-interested mobile devices may be unwilling to cooperate without receiving adequate rewards [55]. Such considerations may be exacerbated in cross-silo federated settings, where competitors should collaborate for a common objective, while they may have local data different in quality (i.e., an organization with rich and high-quality local data would not be willing to participate in a federated learning process and sharing, for free, the acquired final knowledge with other competitors that have contributed much less in the learned model due to scarce-quality data). Furthermore, the revenue generated from the built model will come only afterwards [133]. In this direction, solutions to properly reward participants and attracting data owners with high-quality data, e.g. more conspicuous rewards for participants with higher quality of local data, are emerging (e.g., [55], [133]). Designing effective incentive mechanisms will be fundamental for the spreading of decentralized learning in real-world scenarios.

5.3 Towards Model Heterogeneity and Personalization

As we have seen, in federated settings, different kinds of heterogeneity must be addressed, from system heterogeneity (i.e., device with different resource budgets) to data heterogeneity (i.e., non-IIDness). We highlight an additional facet of heterogeneity that regards the local model architecture: each participant of the learning process can design its own model accordingly to its needs. This degree of freedom would further favour the collaboration among institutions — under the perspective of intellectual property related to the tailored model architecture — and can be also leveraged to favour the inclusion of more resource-constrained edge devices in the learning process. Transfer learning and knowledge distillation are investigated to effectively enabling such independence improvements among participants (e.g., [65]). Besides model heterogeneity, model personalization, i.e. fitting the global model to the participant-specific local data, would represent an additional tool to tackle non-IIDness [62].

5.4 Going beyond Supervised Learning

It is important to underline once more that almost all the cited works in this survey suppose labeled data examples within supervised learning contexts. However, in real federated settings it could not be straightforward to automatically or to manually label data samples; while systems to favour the collection of user-annotated examples are arising (e.g., [78]), the huge amount of unlabeled raw data, that will be produced in the next years at the edge of the network, may not be adequately exploited by only supervised learning techniques. Anyway, opening up to semi-supervised [52], unsupervised or to reinforcement learning approaches would require similar issues in terms of privacy guarantees, heterogeneity, communication efficiency and scalability.

5.5 User Perception of FL Privacy Guarantees

The rising regulations about privacy protection would ideally require the express consent of users for sensitive-data collection and processing. Decentralized learning techniques naturally shape the principles of focused data collection and minimization, on which most of the privacy-related regulations build on as well. However, we might wonder if the average user fully understands the privacy benefits and limitations that come with the design of decentralized learning systems, and in particular with privacy-preserving decentralized learning systems (e.g., differential private decentralized training). In fact, only if the user is aware of the guarantees about privacy protection, she or he can consciously decide whether and which data letting be involved in possible decentralized learning processes. Moreover, different users may value privacy aspects differently, eventually entailing fine granular and user-specific tuning of privacy guarantees, an aspect that has not been thoroughly explored yet. Orthogonally, there is no clear consensus on how to choose privacy parameters (e.g., ϵ for ϵ-DP mechanisms) [28]. Fostering and creating a shared consensus about the adequate level of privacy in collaborative learning systems is another key aspect for the incoming future, as well as fully understanding and addressing the specific privacy preferences of educated users (i.e., users who have full comprehension of the implications of the privacy technology used).

5.6 Fairness and Sources of Bias in Decentralized Learning

The relevant objective of ensuring fairness does not strictly relate to decentralized learning; it is a recognized and well-known issue in traditional ML/DL. However, some unique and peculiar traits of decentralized learning systems open up to new directions for future research. In fact, especially in cross-device settings, practical assumptions and requirements about the (selected) per-round participants can generate bias in the training data, which in turn might make the model unfair, e.g., under-represented groups in training samples may receive lower-quality predictions, or individuals that should be treated similarly by the model receive significantly different outcomes, or again the trained model might show prejudices against some sensitive subgroups of individuals. By going into practical details and consequences, for example, the proposed implementation of FL for Android mobile devices includes in the training rounds only the devices that are (i) connected to unmetered network, (ii) charging, and (iii) that respond within a time-out (also the involved devices have to meet some hardware requirements, i.e., memory); this may lead to sample a biased population of participants. Solutions for more flexible device participation (e.g., [105]) can mitigate such phenomenon. Similar observations raise from other strategies such as prioritizing fast connected devices (e.g., in [117] or [50]). Furthermore, also imbalanced data among nodes can represent a source of bias [26], and this has demonstrated to be more typical of cross-device settings. Another factor that makes fairness challenging in decentralized learning systems lies in the privacy-preserving design of such approaches: usually data are not directly accessible to search for bias in data samples.

5.7 Towards Fully Decentralized Systems at Scale

While cross-device (star-shaped) FL is mature enough to be used in large scale applications [12] (e.g., in the realm of smartphone apps), cross-device fully decentralized solutions have not reached such mature implementations yet. As already highlighted, dealing with peer-to-peer topologies inherently adds layers of complexity with respect to the client-server paradigm; that makes it harder the implementation as well as the theoretical analysis of such systems. A very practical solution may be having a central orchestrating entity that is aware of the current topology status thanks to periodic reports provided by the federation of peers (as in [117]); in this way the orchestrator[21] can determine and dictate the (favourable) peer links to be used in exchanging model updates. In this perspective, in the short-term future research in the field, we expect growing efforts in practical (and maybe more elegant) solutions to dominate the complexity of dynamic large-scale peer-to-peer topologies, as in the case of real cross-device federated scenarios of practical usage, since fully decentralized systems bring, in principle, several advantages with respect to star-shaped solutions (e.g., no need to trust central entities, no server bottlenecks, no unique points of failure). We also note that while communication-efficient strategies can be more easily adapted from star-shaped to fully decentralized systems (e.g., [117]), this may be not so natural for non-IIDness and for privacy guarantees. Furthermore, as far as we know, poisoning has not been investigated considering such topology of participants. In short, the literature about fully decentralized learning is still in its embryonic stages: approaches to ensure formal privacy guarantees (e.g., DP-based approaches and secure aggregation adaptations) and to effectively tackle non-IIDness (e.g., [92]) have still to be thoughtfully explored and investigated before achieving the efficient implementation and deployment of an associated large-scale prototype.

6 CONCLUDING REMARKS

This survey aims at offering a fresh and up-to-date overview of the motivations that are leading to the rising popularity of decentralized learning, by also exemplifying them over a few variegated instances of real-world applications. Most relevantly, the paper proposes an original and relatively simple taxonomy to readily classify baselines and their improvements/extensions for decentralized learning, thus providing a useful guide to and shedding new light on this articulated research area and the emerging frameworks/solutions in the field. The proposed taxonomy has been largely used in the paper as a lens for an in-depth technical analysis of up-to-date contributions in the literature. This analysis has allowed us to highlight the main issues that the surveyed work has addressed and

[21]The orchestrator may also easily dictate the hyperparameters of the model to be trained and of the algorithm to be used.

to identify the primary lessons learned so far; the lessons learned based on our taxonomy-driven analysis also helped us to identify the most relevant open problems and the most promising future directions for research in this challenging, wide, relevant, and rising area.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 308–318.

[2] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. 2018. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems*. 7564–7575.

[3] Alham Fikri Aji and Kenneth Heafield. 2017. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021* (2017).

[4] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. 2018. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235* (2018).

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2-3 (2002), 235–256.

[6] Sean Augenstein, H Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, et al. 2019. Generative Models for Effective ML on Private, Decentralized Datasets. *arXiv preprint arXiv:1911.06679* (2019).

[7] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. 2019. Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems*. 15453–15462.

[8] Evita Bakopoulou, Balint Tillman, and Athina Markopoulou. 2019. A Federated Learning Approach for Mobile Packet Classification. *arXiv preprint arXiv:1907.13113* (2019).

[9] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. 2018. signSGD: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434* (2018).

[10] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. 2016. Gossip training for deep learning. *arXiv preprint arXiv:1611.09726* (2016).

[11] Manuel Blum and Silvio Micali. 1984. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM journal on Computing* 13, 4 (1984), 850–864.

[12] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* (2019).

[13] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1175–1191.

[14] Keith Bonawitz, Fariborz Salehi, Jakub Konečnỳ, Brendan McMahan, and Marco Gruteser. 2019. Federated learning with autotuned communication-efficient secure aggregation. *arXiv preprint arXiv:1912.00131* (2019).

[15] Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. 2018. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics* 112 (2018), 59–67.

[16] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. 2018. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210* (2018).

[17] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. 2020. TiFL: A Tier-based Federated Learning System. *arXiv preprint arXiv:2001.09249* (2020).

[18] Mingqing Chen, Rajiv Mathews, Tom Ouyang, and Françoise Beaufays. 2019. Federated Learning Of Out-Of-Vocabulary Words. *arXiv preprint arXiv:1903.10635* (2019).

[19] Xiangyi Chen, Tiancong Chen, Haoran Sun, Zhiwei Steven Wu, and Mingyi Hong. 2019. Distributed Training with Heterogeneous Data: Bridging Median and Mean Based Algorithms. *arXiv preprint arXiv:1906.01736* (2019).

[20] Yang Chen, Xiaoyan Sun, and Yaochu Jin. 2019. Communication-Efficient Federated Deep Learning With Layerwise Asynchronous Model Update and Temporally Weighted Aggregation. *IEEE Transactions on Neural Networks and Learning Systems* (2019).

[21] Cisco. [n.d.]. Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper. URL https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html. Accessed on April 2020.

[22] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*. 1223–1231.

[23] Whitfield Diffie and Martin Hellman. 1976. New directions in cryptography. *IEEE transactions on Information Theory* 22, 6 (1976), 644–654.

[24] Tung V. Doan, Zhongyi Fan, Giang T. Nguyen, Hani Salah, Dongho You, and Frank HP Fitzek. 2020. Follow Me, If You Can: A Framework for Seamless Migration in Mobile Edge Cloud. *IEEE INFOCOM Workshops* (2020), 1178–11183.

[25] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4829–4837.

[26] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. 2020. Self-Balancing Federated Learning With Global Imbalanced Data in Mobile Systems. *IEEE Transactions on Parallel and Distributed Systems* 32, 1 (2020), 59–71.

[27] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, 7 (2011).

[28] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. 2019. Differential Privacy in Practice: Expose your Epsilons! *Journal of Privacy and Confidentiality* 9, 2 (2019).

[29] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[30] EU. [n.d.]. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. URL https://eur-lex.europa.eu/legal-content/EN/TXT/.

[31] Romano Fantacci and Benedetta Picano. 2020. Federated learning framework for mobile edge computing networks. *CAAI Transactions on Intelligence Technology* 5, 1 (2020), 15–21.

[32] Yingwei Fu, Huaimin Wang, Kele Xu, Haibo Mi, and Yijie Wang. 2019. Mixup Based Privacy Preserving Mixed Collaboration Learning. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 275–2755.

[33] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2018. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866* (2018).

[34] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

[35] Lodovico Giaretta and Šarūnas Girdzijauskas. 2019. Gossip learning: Off the beaten path. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 1117–1124.

[36] Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications* 116 (2018), 1–8.

[37] Meng Hao, Hongwei Li, Guowen Xu, Sen Liu, and Haomiao Yang. 2019. Towards Efficient and Privacy-Preserving Federated Deep Learning. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[38] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[39] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. 2018. Gossiping GANs. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning: DIDL*, Vol. 22.

[40] Valentin Hartmann and Robert West. 2019. Privacy-Preserving Distributed Learning with Secret Gradient Descent. *arXiv preprint arXiv:1906.11993* (2019).

[41] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. FedNAS: Federated Deep Learning via Neural Architecture Search. *arXiv preprint arXiv:2004.08546* (2020).

[42] Chaoyang He, Conghui Tan, Hanlin Tang, Shuang Qiu, and Ji Liu. 2019. Central server free federated learning over single-sided trust social networks. *arXiv preprint arXiv:1910.04956* (2019).

[43] István Hegedűs, Gábor Danner, and Márk Jelasity. 2019. Gossip Learning as a Decentralized Alternative to Federated Learning. In *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 74–90.

[44] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[45] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 603–618.

[46] Wei Hou, Dakui Wang, and Xiaojun Chen. 2020. Generate Images with Obfuscated Attributes for Private Image Classification. In *International Conference on Multimedia Modeling*. Springer, 125–135.

[47] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335* (2019).

[48] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. 2018. LoAdaBoost: Loss-Based AdaBoost Federated Machine Learning on medical Data. *arXiv preprint arXiv:1811.12629* (2018).

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

[50] Jingyan Jiang, Liang Hu, Chenghao Hu, Jiate Liu, and Zhi Wang. 2020. BACombo—Bandwidth-Aware Decentralized Federated Learning. *Electronics* 9, 3 (2020), 440.

[51] Richeng Jin, Yufan Huang, Xiaofan He, Huaiyu Dai, and Tianfu Wu. 2020. Stochastic-Sign SGD for Federated Learning with Theoretical Guarantees. *arXiv preprint arXiv:2002.10940* (2020).

[52] Yilun Jin, Xiguang Wei, Yang Liu, and Qiang Yang. 2020. A Survey towards Federated Semi-supervised Learning. *arXiv preprint arXiv:2002.11545* (2020).

[53] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[54] Michael Kamp, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel. 2018. Efficient decentralized deep learning by dynamic model averaging. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 393–409.

[55] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. 2019. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal* 6, 6 (2019), 10700–10714.

[56] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. 2019. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378* (2019).

[57] Kimon Karras, Evangelos Pallis, George Mastorakis, Yannis Nikoloudakis, Jordi Mongay Batalla, Constandinos X. Mavromoustakis, and Evangelos K. Markakis. 2020. A Hardware Acceleration Platform for AI-Based Inference at the Edge. *Circuits Syst. Signal Process.* 39, 2 (2020), 1059–1070.

[58] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. 2018. On-device federated learning via blockchain and its latency analysis. *arXiv preprint arXiv:1808.03949* (2018).

[59] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[60] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[61] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[62] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. 2020. Survey of Personalization Techniques for Federated Learning. *arXiv preprint arXiv:2003.08673* (2020).

[63] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[64] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. 2019. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6341–6345.

[65] Daliang Li and Junpu Wang. 2019. FedMD: Heterogenous Federated Learning via Model Distillation. *arXiv preprint arXiv:1910.03581* (2019).

[66] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen. 2019. Abnormal client behavior detection in federated learning. *arXiv preprint arXiv:1910.09933* (2019).

[67] Tian Li, Zaoxing Liu, Vyas Sekar, and Virginia Smith. 2019. Privacy for Free: Communication-Efficient Learning with Differential Privacy Using Sketches. *arXiv preprint arXiv:1911.00972* (2019).

[68] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[69] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2018. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127* (2018).

[70] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. FedDANE: A Federated Newton-Type Method. *arXiv preprint arXiv:2001.01920* (2020).

[71] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[72] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*. 5330–5340.

[73] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* (2020).

[74] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. 2017. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887* (2017).

[75] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. 2019. Edge-Assisted Hierarchical Federated Learning with Non-IID Data. *arXiv preprint arXiv:1905.06641* (2019).

[76] Menghan Liu, Haotian Jiang, Jia Chen, Alaa Badokhon, Xuetao Wei, and Ming-Chun Huang. 2016. A collaborative privacy-preserving deep learning system in distributed mobile environment. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 192–197.

[77] Wei Liu, Li Chen, Yunfei Chen, and Wenyi Zhang. 2020. Accelerating Federated Learning via Momentum Gradient Descent. *IEEE Transactions on Parallel and Distributed Systems* 31, 8 (2020), 1754–1766.

[78] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. 2020. FedVision: An Online Visual Object Detection Platform Powered by Federated Learning. *arXiv preprint arXiv:2001.06202* (2020).

[79] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to Federated Learning: A Survey. *arXiv preprint arXiv:2003.02133* (2020).

[80] Evangelos K. Markakis, Kimon Karras, Nikolaos Zotos, Anargyros Sideris, Theoharris Moysiadis, Angelo Corsaro, George Alexiou, Charalabos Skianis, George Mastorakis, Constandinos X. Mavromoustakis, and Evangelos Pallis. 2017. EXEGESIS: Extreme Edge Resource Harvesting for a Virtualized Fog Environment. *IEEE Commun. Mag.* 55, 7 (2017), 173–179.

[81] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. 2016. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629* (2016).

[82] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private language models without losing accuracy. *arXiv preprint arXiv:1710.06963* (2017).

[83] H Brendan McMahan and Matthew Streeter. 2010. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908* (2010).

[84] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 691–706.

[85] Jed Mills, Jia Hu, and Geyong Min. 2019. Communication-Efficient Federated Learning for Wireless Edge Intelligence in IoT. *IEEE Internet of Things Journal* (2019).

[86] Akinori Mitani, Abigail Huang, Subhashini Venugopalan, Greg S Corrado, Lily Peng, Dale R Webster, Naama Hammel, Yun Liu, and Avinash V Varadarajan. 2020. Author Correction: Detection of anaemia from retinal fundus images via deep learning. *Nature Biomedical Engineering* 4, 2 (2020), 242–242.

[87] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[88] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust de-anonymization of large datasets (how to break anonymity of the Netflix prize dataset). *University of Texas at Austin* (2008).

[89] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910* (2018).

[90] Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed. 2020. Federated Learning for Wireless Communications: Motivation, Opportunities, and Challenges. *IEEE Communications Magazine* 58, 6 (2020), 46–51.

[91] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[92] Kenta Niwa, Noboru Harada, Guoqiang Zhang, and W Bastiaan Kleijn. 2020. Edge-consensus Learning: Deep Learning on P2P Networks with Nonhomogeneous Data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 668–678.

[93] State of California Department of Justice. [n.d.]. California Consumer Privacy Act (CCPA). URL https://oag.ca.gov/privacy/ccpa. Accessed on May 2020.

[94] U.S. Department of Health & Human Services. [n.d.]. The HIPAA Privacy Rule. URL https://www.hhs.gov/hipaa/for-professionals/privacy/index.html. Accessed on May 2020.

[95] Trishan Panch, Peter Szolovits, and Rifat Atun. 2018. Artificial intelligence, machine learning and health systems. *Journal of global health* 8, 2 (2018).

[96] Stephen R Pfohl, Andrew M Dai, and Katherine Heller. 2019. Federated and Differentially Private Learning for Electronic Health Records. *arXiv preprint arXiv:1911.05861* (2019).

[97] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. 2018. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security* 13, 5 (2018),

1333–1345.

[98] Tran Thi Phuong et al. 2019. Privacy-preserving deep learning via weight transmission. *IEEE Transactions on Information Forensics and Security* 14, 11 (2019), 3003–3015.

[99] Maarten G Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar. 2019. Split Learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115* (2019).

[100] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated Learning for Emoji Prediction in a Mobile Keyboard. *arXiv preprint arXiv:1906.04329* (2019).

[101] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. 2020. Adaptive Federated Optimization. *arXiv preprint arXiv:2003.00295* (2020).

[102] Sashank J Reddi, Jakub Konečnỳ, Peter Richtárik, Barnabás Póczós, and Alex Smola. 2016. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879* (2016).

[103] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. 2019. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. *arXiv preprint arXiv:1909.13014* (2019).

[104] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. 2019. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731* (2019).

[105] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. 2020. Towards Flexible Device Participation in Federated Learning for Non-IID Data. *arXiv preprint arXiv:2006.06954* (2020).

[106] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems* (2019).

[107] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Sparse binary compression: Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[108] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. 2020. Federated Learning with Cooperating Devices: A Consensus Approach for Massive IoT Networks. *IEEE Internet of Things Journal* (2020).

[109] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[110] Ohad Shamir, Nati Srebro, and Tong Zhang. 2014. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*. 1000–1008.

[111] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. 2019. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145* (2019).

[112] Jinhyun So, Basak Guler, and A Salman Avestimehr. 2020. Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning. *arXiv preprint arXiv:2002.04156* (2020).

[113] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. 2018. Human Activity Recognition Using Federated Learning. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE, 1103–1111.

[114] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[115] Nikko Strom. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*.

[116] Gábor J Székely, Maria L Rizzo, Nail K Bakirov, et al. 2007. Measuring and testing dependence by correlation of distances. *The annals of statistics* 35, 6 (2007), 2769–2794.

[117] Zhenheng Tang, Shaohuai Shi, and Xiaowen Chu. 2020. Communication-efficient decentralized learning with sparsification and adaptive peer selection. *arXiv preprint arXiv:2002.09692* (2020).

[118] Zeyi Tao and Qun Li. 2018. esgd: Communication efficient distributed deep learning on the edge. In *{USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 18)*.

[119] Chandra Thapa, MAP Chamikara, and Seyit Camtepe. 2020. SplitFed: When Federated Learning Meets Split Learning. *arXiv preprint arXiv:2004.12088* (2020).

[120] Aleksei Triastcyn and Boi Faltings. 2019. Federated Learning with Bayesian Differential Privacy. *arXiv preprint arXiv:1911.10071* (2019).

[121] Aleksei Triastcyn and Boi Faltings. 2019. Improved Accounting for Differentially Private Learning. *arXiv preprint arXiv:1901.09697* (2019).

[122] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. 1–11.

[123] Praneeth Vepakomma, Otkrist Gupta, Abhimanyu Dubey, and Ramesh Raskar. 2019. Reducing leakage in distributed deep learning for sensitive health data. *arXiv preprint arXiv:1812.00564* (2019).

[124] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. 2020. Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1698–1707.

[125] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221.

[126] Wikipedia. [n.d.]. Facebook–Cambridge Analytica data scandal. URL https://en.wikipedia.org/wiki/Facebook\T1\textendashCambridge_Analytica_data_scandal. Accessed on May 2020.

[127] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).

[128] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. SLSGD: Secure and Efficient Distributed On-device Machine Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

[129] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[130] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).

[131] Xin Yao, Tianchi Huang, Rui-Xiao Zhang, Ruiyu Li, and Lifeng Sun. 2019. Federated Learning with Unbiased Gradient Aggregation and Controllable Meta Updating. *arXiv preprint arXiv:1910.08234* (2019).

[132] Chun-Hsien Yu, Chun-Nan Chou, and Emily Chang. 2019. Distributed Layer-Partitioned Training for Privacy-Preserved Deep Learning. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 343–346.

[133] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. 2020. A Fairness-aware Incentive Scheme for Federated Learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 393–399.

[134] Qian Yu, Songze Li, Netanel Raviv, Seyed Mohammadreza Mousavi Kalan, Mahdi Soltanolkotabi, and Salman Avestimehr. 2019. Lagrange Coded Computing: Optimal Design for Resiliency, Security, and Privacy. In *International Conference on Artificial Intelligence and Statistics (AISTATS 2019)*.

[135] Zhengxin Yu, Jia Hu, Geyong Min, Haochuan Lu, Zhiwei Zhao, Haozhe Wang, and Nektarios Georgalas. 2018. Federated learning based proactive content caching in edge computing. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.

[136] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. 2018. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*. 9793–9803.

[137] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. 2019. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2224–2287.

[138] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional mechanism: regression analysis under differential privacy. *arXiv preprint arXiv:1208.0219* (2012).

[139] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[140] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. iDLG: Improved Deep Leakage from Gradients. *arXiv preprint arXiv:2001.02610* (2020).

[141] Lingchen Zhao, Shengshan Hu, Qian Wang, Jianlin Jiang, Chao Shen, and Xiangyang Luo. 2019. Shielding Collaborative Learning: Mitigating Poisoning Attacks through Client-Side Detection. *arXiv preprint arXiv:1910.13111* (2019).

[142] Lingchen Zhao, Qian Wang, Qin Zou, Yan Zhang, and Yanjiao Chen. 2019. Privacy-preserving collaborative deep learning with unreliable participants. *IEEE Transactions on Information Forensics and Security* 15 (2019), 1486–1500.

[143] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. 2019. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing* 115 (2019), 213–237.

[144] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, and Shui Yu. 2019. Multi-Task Network Anomaly Detection using Federated Learning. In *Proceedings of the Tenth International Symposium on Information and Communication Technology*. 273–279.

[145] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).

[146] Jun Zhou, Zhenfu Cao, Xiaolei Dong, and Xiaodong Lin. 2015. PPDM: A privacy-preserving protocol for cloud-assisted e-healthcare systems. *IEEE Journal of Selected Topics in Signal Processing* 9, 7 (2015), 1332–1344.

[147] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. 2019. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* 107, 8 (2019), 1738–1762.

[148] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*. 14747–14756.