



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

## ARCHIVIO ISTITUZIONALE DELLA RICERCA

### Alma Mater Studiorum Università di Bologna Archivio istituzionale della ricerca

Decentralised Learning in Federated Deployment Environments

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

*Published Version:*

Bellavista, P., Foschini, L., Mora, A. (2021). Decentralised Learning in Federated Deployment Environments. ACM COMPUTING SURVEYS, 54(1), 1-38 [10.1145/3429252].

*Availability:*

This version is available at: <https://hdl.handle.net/11585/851144> since: 2022-02-01

*Published:*

DOI: <http://doi.org/10.1145/3429252>

*Terms of use:*

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).  
When citing, please refer to the published version.

(Article begins on next page)

This is the final peer-reviewed accepted manuscript of:

**Bellavista, Paolo and Foschini, Luca and Mora, Alessio, Decentralised Learning in Federated Deployment Environments: A System-Level Survey (2021), ACM COMPUTING SURVEYS. vol. 54, n. 1, issn 0360-0300**

The final published version is available online at:  
<https://dx.doi.org/10.1145/3429252>

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

*This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>)*

***When citing, please refer to the published version.***

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

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2018 Association for Computing Machinery.

XXXX-XXXX/2018/12-ART \$15.00

<https://doi.org/10.1145/1122445.1122456>

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

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

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

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

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

94 We are aware of the rich existing survey literature in the field and in particular of the valuable  
95 [68], [129], [73], and [147] papers. However, we claim that we are providing the readers with a  
96 valuable and differentiated contribution if compared with those surveys primarily because of the  
97 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 Analytica 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<sup>1</sup>, 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

---

<sup>1</sup>At least until the time this survey has been written.

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

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

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

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

190 Confirming its versatility, decentralized learning has been also applied to network traffic classifi-  
191 cation, anomaly detection, and VPN traffic recognition tasks, while preserving appropriate privacy  
192

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

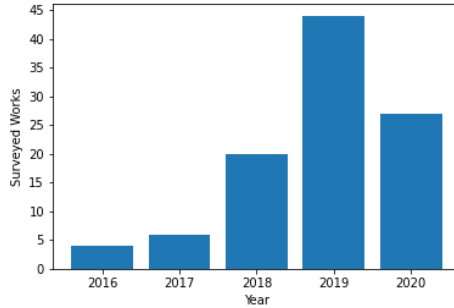


Fig. 1. The histogram reports the number of papers about decentralized learning per year, covered by this survey, by showing the increasing relevance of decentralized learning in the literature.

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



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

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

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

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

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

### 287 3.2 A Taxonomy for Decentralized Learning Systems

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

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



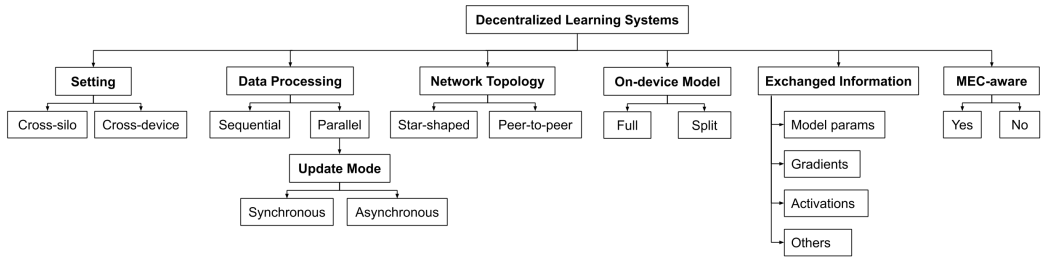


Fig. 2. Our taxonomy for decentralized learning systems.

**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<sup>4</sup>.

**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

<sup>4</sup>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,  $\eta$  is the learning rate. Note the common initialization of model parameters  $w_0$ .

**Server executes:**

initialize  $w_0$

**for** each round  $t = 1, 2, 3, \dots$

$m \leftarrow \max(C \times K, 1)$

$S_t \leftarrow$  (random set of  $m$  clients)

**for** each client  $k \in S_t$  **in parallel**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

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

**ClientUpdate**( $k, w$ )

$\mathcal{B} \leftarrow$  (split  $\mathcal{D}_k$  into batches of size  $B$ )

**for** each local epoch  $e$  from 1 to  $E$

**for** batch  $b \in \mathcal{B}$

$w \leftarrow w - \eta \nabla \ell(w; b)$

**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  $O(\frac{1}{T})$ , 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

442 of knowledge distillation, namely co-distillation [44], [4]. In a nutshell, each device (the student)  
 443 stores its model outputs, i.e. a set of logit values normalized via softmax function, from which it  
 444 derives per-label mean logit vectors, and periodically uploads such local-average logit vectors to  
 445 the aggregator. The server produces the per-label global-average logit vector by averaging the  
 446 contributions of all the participants in that round, and broadcasts such aggregation to the federation;  
 447 each device treats the received per-label global-average logit vector as the teacher's output, and  
 448 locally calculates the distillation regularizer. It is straightforward to note that exchanging logit-  
 449 vector (local or global averaged, whether they are upload or download parameters), in place of  
 450 model parameters or gradients, reduces the per-round communication cost with respect to FedAvg:  
 451 the dimension of logit-vectors depends on the number of labels, and not on the number of model  
 452 parameters.

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

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

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

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

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

483 **3.3.2 Peer-to-peer baselines.** In star-shaped FL, the coordination server orchestrates the commu-  
 484 nication rounds; it iteratively broadcasts the current model state to the participants and gathers  
 485 the locally computed updates to produce the next-generation model by aggregation. Although  
 486 leveraging a client-server architecture permits to ignore topology-related issues, FL presents two  
 487

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

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

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

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

514 In the continuation of this subsection, we present examples of baseline algorithms that consider  
 515 fixed-topology and undirected graphs — most common assumptions. The first work, BrainTorrent  
 516 [104], targets cross-silo federated settings, while the subsequently presented ones also embrace the  
 517 cross-device setting [43] [50] [108].

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

529 Gossip-based protocol for distributed learning has been explored in the datacenter setting as  
 530 alternative to the parameter-server approach (e.g., [10], [39]). Inspired from them, Gossip Learning  
 531 (GL) has been proposed in [43] for Cross-device federated settings. In the baseline GL algorithm,  
 532 starting from a common initialization, each node sends its local model to a randomly selected peer,  
 533 which firstly merges (e.g., by averaging and weighing the average according to an age parameter  
 534 associated with the freshness of the models) the received model with its current parameters, then  
 535 updates the resulting model by exploiting its private dataset, and the process repeats. In a nutshell,  
 536 there could be different models scattered across the network of peers, with each one of these models  
 537 taking random walks (in the network) and being updated when visiting a new node. Typically, the  
 538

**Algorithm 2:** Consensus FedAvg algorithm

$N_k^-$  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,  $\eta$  is the learning rate.

**Participant  $k$  executes:**

```

544 initialize  $w_0^k$ 
545 for each round  $t = 1, 2, 3, ..$ 
546     receive  $\{w_t^i\}_{i \in N_k^-}$ 
547      $\psi_t^k \leftarrow w_t^k$ 
548     for all devices  $i \in N_k^-$ 
549          $\psi_t^k \leftarrow \psi_t^k + \zeta_t \alpha_{t,i} (w_t^i - w_t^k)$ 
550      $w_{t+1}^k = \text{ModelUpdate}(\psi_t^k)$ 
551     send( $w_{t+1}^k$ ) to neighbors

```

**ModelUpdate**( $\psi_t^k$ )

```

553      $\mathcal{B} \leftarrow$  (split  $\mathcal{D}_k$  into batches of size  $B$ )
554     for batch  $b \in \mathcal{B}$ 
555          $\psi_t^k \leftarrow \psi_t^k - \eta \nabla \ell(\psi_t^k; b)$ 
556      $w_t^k \leftarrow \psi_t^k$ 
557     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,  $\psi^k$ . 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



589 is obtained, at round  $t$ , from the neighbor contributions, that is:  $\psi_t^k = w_t^k + \zeta_t \sum_{i \in N_k^-} \alpha_{k,i} (w_t^i - w_t^k)$ ,  
 590 where  $\zeta_t$  is the “consensus step size” and the mixing weights  $\alpha_{k,i}$  are chosen, similarly to FedAvg,  
 591 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  $i$ .  
 592

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

## 597 4 DECENTRALIZED LEARNING SOLUTIONS: A SYSTEM-LEVEL ANALYSIS

598 Decentralized learning decouples by design the ability to learn a predictive ML/DL model from  
 599 the direct access to raw data and meets the rising urge of ensuring privacy guarantees to the data  
 600 owners while still being able to distill useful information for the community. However, as already  
 601 pointed out in this survey, diverse challenges emerge. Chief among them, privacy is not completely  
 602 secured by means of just disclosing ephemeral updates (e.g., gradients, model parameters) or  
 603 meta-level information, as well as the communication efficiency is of paramount importance in  
 604 cross-device federated settings. Furthermore, having the raw data (massively) distributed and/or  
 605 unbalanced among participants naturally implies dealing with non-IIDness. An additional factor  
 606 to be addressed is the heterogeneity of devices’ resources in cross-device settings. Moreover, the  
 607 design of decentralized learning approaches opens up to new possibilities for attackers, since  
 608 learners actively participate in the training process, e.g. forcing information leakage from other  
 609 participants or trying to influence the behaviour of the system. These are the most investigated  
 610 issues in literature so far, but other less crucial aspects and challenges are rising and taking the  
 611 scene while effective solutions for the urgent aspects permit to already apply decentralized learning  
 612 in real scenarios. In this section, we discuss the systems in the literature that aim at solving the  
 613 above mentioned issues, i.e. communication efficiency, privacy, non-IIDness, device heterogeneity,  
 614 and poisoning defense, classifying them by our taxonomy (see Table 1).  
 615

616 Let us note that, in the following sub-sections, we will use the taxonomy definitions and terms  
 617 introduced previously in this survey; where not possible or convenient, we explain in-line the  
 618 specific meaning of the employed definitions/terms/symbols.  
 619

### 620 4.1 Improving Communication Efficiency

621 The communication efficiency in decentralized learning can be addressed from different perspec-  
 622 tives. In the first place, decentralized optimization algorithms are usually designed to allow for  
 623 multiple local training iteration between communication rounds to reduce the total communication  
 624 cost of the training process (e.g., [81], [54]); in synchronous star-shaped federated learning the  
 625 number of participants selected per round is typically limited (e.g., [81]), as well as in peer-to-peer  
 626 topology the number of neighbours to scatter the updates to is bounded (e.g. bounded to 1 such as  
 627 in GL [43] or in [117]). Stream compression (e.g., by encoding, quantization and/or sparsification  
 628 of updates) is typically employed to reduce the per-round communication cost [61] [16] [103]  
 629 [106] [118] [85] [67] [51] [117]. Furthermore, specific strategies can be crafted accordingly to the  
 630 peculiarities of the model to train (e.g., by introducing asynchrony between the updating of the  
 631 neural-network parameters belonging to shallower/deeper layers [20]). Stream compression has  
 632 been mostly explored in star-shaped federated learning, but similar solutions may be easily adapted  
 633 in peer-to-peer topology. An orthogonal approach is to improve the communication efficiency by  
 634 reducing the total communication rounds needed for the model convergence (e.g., implementing  
 635 distributed variants of SGD optimizers [85] [77] [108]). Or again, communication-efficiency can be  
 636  
 637



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<sup>st</sup> moments,  $v$  2<sup>nd</sup> Adam moments,  $c$  control variates,  $d$  GD momentum,  $t$  time stamps,  $res\_info$  resource information,  $L$  loss function value,  $\rho$  the Lipschitz parameter of the loss function,  $\beta$  the smoothness parameter of the loss function,  $\tau^*$  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	✓			✓	✓		$w, g, \rho, \beta,$ $L, res\_info$	$w, \tau^*$	×
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
RD	SLSGD [128]	2019	device	Full	✓		✓	✓		w	w	×
	FoolsGold [33]	2018	device	Full	✓		✓	✓		g	w	×
	L. Zhao et al. [141]	2019	device	Full	✓		✓	✓		w	w	×
	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<sup>6</sup> 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

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

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

747 Building on their previous Sparse Binary Compression (SBC) [107] technique that targets the  
 748 traditional distributed setting, in [106] authors specifically design a compression framework for  
 749 cross-device federated settings. The proposed Sparse Ternary Compression (STC) compresses both  
 750 the upstream and the downstream communication with respect to the baseline FedAvg while  
 751 improving the robustness to non-IID data as well as to partial client participation. In addition  
 752 to experimentally confirming the already known weakness of vanilla FedAvg in presence of het-  
 753 erogeneous data, authors also show poor model accuracy with aggressive quantization schemes,  
 754 such as SignSGD<sup>7</sup> [9], in non-IID scenarios. Conversely,  $top_p\%$  sparsification, i.e. dropping all but  
 755 the  $p$  fraction of updates with the highest magnitude, suffers least from heterogeneous data. This  
 756 observation leads the design of the proposed compression scheme for the upstream communi-  
 757 cation in FL. As happens in SBC, STC exploits (i)  $top_p\%$  sparsification of weight deltas (i.e., the  
 758 difference between the global model and the local model), (ii) local residual accumulation<sup>8</sup>, (iii)  
 759 binary quantization of the  $top_p\%$  elements<sup>9</sup> and (iv) encoding (to losslessly compress the distance  
 760 between the non-zero elements of the sparse weight-update) to reduce the amount of data to be  
 761 sent from participants to the server. It is worth to highlight once more that this strategy alone  
 762 does not affect the downstream communication. In this regard, authors observe that, although  
 763 clients-to-server updates are sparse, the server-to-clients update essentially becomes dense as  
 764 the participation rate, i.e. the fraction of participants involved in each round, exceeds the inverse  
 765 sparsity, i.e. the inverse of the hyperparameter that rules the sparsification. In fact, in the worst case,  
 766 the number of non-zero elements in the aggregate (the sum) of clients-to-server updates grows  
 767 linearly with the number of participating clients. The dense nature of server-to-clients updates  
 768 prevent an effective compression. Therefore, they propose to apply their STC algorithm also to the  
 769 aggregated updates at server side, hence the server maintains a residual as well. However, the partial  
 770 client participation in each round of FL prevents a straightforward application of STC at server-side:  
 771 STC sparsifies and compresses weight deltas, and, considering that not all the participants are  
 772 involved in every round, some participants could not recover the updated weights from the received  
 773 (compressed) delta, since they may not have participated to the previous round(s). The solution  
 774

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

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

780 <sup>9</sup>The result of the sparse weight-update binarization is a ternary tensor containing values  $-\mu, 0, \mu$  with  $\mu$  being the mean  
 781 of the  $top_p\%$  weight-updates in absolute value. STC sets all the positive non-zeroed elements to  $\mu$  and all the negative  
 782 non-zeroed elements to  $-\mu$ . Note that in SBC the resulting sparse tensor is binary instead, and the algorithm is slightly  
 783 different; they independently compute the mean of all non-zeroed positive and all non-zeroed negative weight-updates; if  
 784 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  $\tau$  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<sup>10</sup> – 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  $\tau_1$  local updates, edge servers gather the updates of the participants in their proximity to produce the aggregated models of their locality; each  $\tau_2$  edge-level aggregations, the cloud updates the global model (hence each  $\tau_1 \tau_2$  local iterations). It is worth noting that if  $\tau_2$  is equal to 1, the HierFAVG corresponds to the traditional FedAvg, while, intuitively, with  $\tau_2$  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  $\psi_t^k$  to run local training, the participant  $k$  feeds back  $\psi_t^k$  to the same neighbors.

<sup>10</sup>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.

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

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

## 859 4.2 Protecting Privacy

860 It may be believed that sharing gradients, model updates or meta-level information (such as  
 861 outputs of layers in neural-networks) in place of raw data ensures privacy protection. However, it  
 862 has been demonstrated that gradients exchanged during the distributed training process do leak  
 863 information about the training data [148] [140] [40] [97] [89] [45] as well as model updates [84]  
 864 [89] – even though it may be preferable to exchange model weights instead of gradients under a  
 865 privacy-preserving perspective [98] – and activations [25] [132].

866 The literature about protecting privacy in decentralized learning comprises diverse approaches;  
 867 differentially-private mechanisms [34] [82] can be employed during the distributed training process  
 868 to mask updates at the cost of reduced model accuracy [7], and relaxations of traditional Differential  
 869 Privacy (DP) can be leveraged to inject less noise [120], limiting the incurred performance degrada-  
 870 tion. Data-augmentation [32] and obfuscation [46] techniques can be used in visual application to  
 871 prevent reconstruction of images in the training set. Multi-party secure aggregation [13] [112] and  
 872 similar techniques [40] can hide the individual contributions to the aggregator, finding its main  
 873 utility in star-shaped federated learning, but producing non-negligible overheads. Additively homo-  
 874 morphic encryption also allows the aggregator to sum updates, thus ensuring the inscrutability of  
 875 single contributions [97] while not degrading model accuracy but increasing communication cost.  
 876 Combinations of DP-mechanisms with secure aggregation and additively homomorphic encryption  
 877 are also explored [122] [37] to balance the weaknesses of such techniques. Minimizing distance  
 878 correlation between raw data and activations (at cut layer) [123] and step-wise activation functions  
 879 [132] are used to prevent the invertibility from intermediary representations in the context of  
 880

881 <sup>11</sup>The negotiation phase, from an high-level perspective, can be thought to be similar to the approach of [70].  
 882



883 privacy-preserving Split Learning.

884  
885 The first works enforcing participant-level  $(\epsilon, \delta)$ -DP [29] in federated settings are most notably  
886 [34] and [82]. The aim, common to both the works, is to ensure that a model trained with FedAvg  
887 does not reveal whether a certain participant has been involved during the decentralized training  
888 process, balancing the trade-off between privacy loss and model performance. It is worth high-  
889 lighting that the proposed solutions protect the whole client's dataset differently from [1] where a  
890 single data point's contribution in the trained model is protected.

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

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

914 In [120], authors allocate a tighter privacy budget for guaranteeing client-level DP and instance-  
915 level DP, i.e. less noise to reach the same privacy guarantee, also improving the accuracy of the  
916 trained model. They employ a relaxation of traditional DP, in this case Bayesian DP (BDP) [121], by  
917 making two assumptions (i) stationary data distribution and (ii) datasets with unchangeable samples.  
918 Authors also use a Bayesian accounting method instead of state-of-the-art moments accountant  
919 [1] thanks to the assumption that data come from a particular distribution and not all the data  
920 are equally likely; this observation can lead to sharper privacy loss bounds with BDP in federated  
921 setting. Besides the proposed use of BDP, to limit the noise added to guarantee both instance-  
922 level and client-level DP, the noise to be added by the server for client-level DP is "re-counted"  
923 considering the injected noise during the on-device gradient descent. They call this approach joint  
924 accounting. However, a limitation emerges: joint accounting is only usable for FedSGD algorithm,  
925 not for FedAvg (because the possible multiple local iterations in FedAvg, hence multiple noisy steps,  
926 may influence the point at which the gradient is computed: a different gradient distribution can  
927 arise or the total noise variance can be underestimated).

928  
929  
930 <sup>12</sup>The sensitivity is calculated by the server in each communication round.

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

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

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

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

980



981 Firstly, participants are randomly divided in  $L$  groups, with each group being composed of  $N_l$   
 982 participants. The set of participants in group  $l$  is referred as  $U_l$ . The process involves  $L$  stages,  
 983 and Turbo-Agg adopts a circular and sequential strategy in its simplest version: in each stage  
 984 only one group is involved; the output produced from a group in a certain stage is the input for  
 985 the next group<sup>13</sup>. Ignoring for a moment the possibility of dropout, in each stage, the participant  
 986  $i$  in group  $l$  masks its update  $x_i^{(l)}$  with a random vector  $u_i^{(l)}$  being known (and communicated)  
 987 only by the honest server, similarly to what happens in SECAGG. To be secure against server-  
 988 participants collusion, learner  $i$  additionally masks its update with another random vector  $r_{i,j}^{(l)}$ ,  
 989 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  
 990  $l + 1$ , 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  
 991 cooperatively computed, group by group, and can be summarized thanks to the recursive relation  
 992  $\tilde{s}_i^{(l)} = \frac{1}{N_{l-1}} \sum_{j \in [N_{l-1}]} \tilde{s}_j^{(l-1)} + \sum_{j \in [U_{l-1}]} \tilde{x}_{j,i}^{(l-1)}$  with  $\tilde{s}_i^{(l)}$  that is a variable locally held by each partici-  
 993 pant  $i$  in group  $l > 1$ , and that represents the aggregated masked updates from the previous group<sup>14</sup>.  
 994 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  
 995 the group  $l + 1$ . A final aggregation step is necessary to preserve the privacy of the participants in  
 996 group  $L$  at the stage  $L$ ; an additional group (referred as *final*), in fact, is randomly composed (for  
 997 example, among the survived learners) with each participant aggregating the contributions coming  
 998 from the group  $L$ , and sending the results to the server. Specifically, participants  $j$  in the *final* group  
 999 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  
 1000 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,  
 1001 in case of participant dropouts the protocol will fail, since, for example, the random vectors  $r$   
 1002 cannot be cancelled out. To this end, authors propose to employ Lagrange coding [134] to allow  
 1003 participants of group  $l$  to recover the missing contributions from group  $l - 1$ , and to compute the  
 1004 partial aggregation anyway. Being concrete and redirecting to the full paper [112] and to [134] for  
 1005 theoretical detail, each participant has to send to each participant  $j$  in group  $l + 1$  two additional  
 1006 (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  
 1007 strategy allow each learner in group  $l + 1$  to reconstruct the vector  $\{\tilde{s}_i^{(l)}\}_{i \in N_l}$  starting from at least  
 1008  $N_l$  evaluations (i.e.  $\bar{s}_i^{(l)}$  and  $\bar{x}_{i,j}^{(l)}$ ) from the previous stage. Therefore, since each participant send  
 1009 two evaluations to the learners in the next group, this redundancy permits to tolerate up to half of  
 1010 learners dropping.  
 1011

1012 It is worth noting that, although SECAGG and its variant Turbo-Aggregate explicitly targets  
 1013 star-shaped networks of learners, they are suitable for fully decentralized networks, i.e. peer-to-peer  
 1014 topologies, with one peer (or more) working as aggregator.  
 1015

1016 An alternative to SECAGG for star-shaped FL frameworks is represented by Additively Ho-  
 1017 momorphic Encryption; since such technique guarantees the additivity of multiple ciphertexts,  
 1018 the server can perform the aggregation without the need of seeing the updates in clear. In [37],  
 1019 authors propose to use a symmetric additively homomorphic encryption called PPDM [146] for its  
 1020 efficiency, combining it with Laplacian mechanism for DP in order to neutralize collusion between  
 1021 compromised users and malicious server. They show drastically reduced communication overhead  
 1022 with similar solution [97], that employs paillier encryption instead.  
 1023

1024 In [122], authors combine multi-party computation (MPC) via Threshold Homomorphic Encryp-  
 1025 tion and Differential Privacy to balance their respective weaknesses; in fact, applying DP to provide  
 1026

1027 <sup>13</sup>Since only one group is active per stage, for ease of notation, group and stage are referred both with the index  $l$ .

1028 <sup>14</sup>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  $\epsilon$ -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.

Considering 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  $\epsilon$ -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<sup>15</sup>. 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].

<sup>15</sup>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.

1079 In [51], authors combine communication efficiency, privacy guarantees and resilience to malicious  
 1080 participants under non-IID data distribution. They consider a star-shaped synchronous collaborative  
 1081 learning framework in which participants and server exchange (aggressively compressed) gradients  
 1082 instead of model parameters. The proposed algorithms use as baseline the SignSGD [9] algorithm  
 1083 with majority vote, that, however, does not explicitly and formally address privacy protection  
 1084 of participants and that has been shown to fail to converge when the data on different learners  
 1085 are heterogeneous [19] [106]. In particular, to deal with non-IID data, authors first propose a  
 1086 variation of SignSGD, namely *sto-sign*, that applies a two-level stochastic quantization on locally  
 1087 computed gradients, and then only transmits the signs of such quantized values. Additionally, *dp-*  
 1088 *sign*, a differentially private version of *sto-sign*, is designed to ensure formal privacy guarantees for  
 1089 participants involved in the training. Authors theoretically relate the Byzantine<sup>16</sup> resilience, i.e. the  
 1090 number of Byzantine workers that can be tolerated without harming the convergence guarantees, of  
 1091 their proposed algorithms to the heterogeneity of local datasets. Authors also propose an extension  
 1092 of their algorithms which takes account for residual error on server side and uses it to correct the  
 1093 majority vote. The convergence of the proposed algorithms is established theoretically.

1094 With respect to just sending the quantized updates in clear, the SECAGG[13] protocol leads to  
 1095 a bandwidth expansion<sup>17</sup> that is less than 2x while ensuring reliability of the secure aggregation  
 1096 to dropping or collusion of a fraction of users. However, in [14], authors critically observe some  
 1097 limitations of a straightforward combination of SECAGG and compression techniques; chief among  
 1098 them (i) quantizing to a fixed point representation requires selecting the clipping range  $[-c, c]$   
 1099 a priori that may be challenging to establish or may lead to poor approximations if the clipping  
 1100 range is not large enough, and (ii) the SECAGG modulus is chosen to be  $K = nk$  to represent all  
 1101 possible aggregated vectors without overflow (for example, if clients are  $2^{10}$  the SECAGG modulus  
 1102 are 10 bits wider than they would be without accounting for secure aggregation) dominating the  
 1103 communication cost introduced by SECAGG – the bandwidth expansion determined by secret  
 1104 sharing and cryptography is much less influential. The scope of [14] is to propose a recipe for an  
 1105 auto-tuning (observation (i)) communication-efficient (observation (ii)) secure aggregation. The key  
 1106 idea is to avoid clipping at client-side but instead quantizing over an unbounded range according to  
 1107 a quantization bin size  $b$  that is dynamically and tightly adjusted by the server (and communicated  
 1108 round by round) according to the distribution of the entries of the sum relative to the previous  
 1109 round, and then locally applying the *mod k* operation instead of clipping; the server can compute  
 1110 a tight bin size  $b$  exploiting the assumption that the entries of the sum fit a normal distribution  
 1111 thanks to a random rotation that is locally performed by the participants (before quantizing) to  
 1112 their updates.

#### 1113 4.4 Addressing non-IIDness

1115 As empirically shown by [81], carefully tuning the number of local epochs is crucial in FedAvg  
 1116 since during additional on-device iterations – less frequent synchronization among participants –  
 1117 local models can significantly drift apart from the global model potentially preventing convergence.  
 1118 Such an issue is exacerbated when considering statistically heterogeneous data from different par-  
 1119 ticipants [81] [145] [107] [47] – realistic assumption especially in cross-device federated settings.  
 1120 Data sharing and data augmentation techniques have been demonstrated to effectively alleviate  
 1121 the impact of non-IIDness at the cost of less decentralization [145] [131] [49]. Another major line  
 1122 of works tackles the problem by directly limiting the drift of the model’s objective function by

1123 <sup>16</sup>A Byzantine participant may transmit arbitrary information. Authors of [51] assume that such Byzantine participants  
 1124 upload the opposite signs (the opposite sign of each entry) of the true gradients, with the true gradients being the average  
 1125 gradients of all the normal workers (hence, it is supposed that the attackers know such quantities).

1126 <sup>17</sup>1.73x bandwidth expansion considering  $2^{10}$  participants (i.e.,  $n = 2^{10}$ ) and 16 bit fixed point representation (i.e.,  $k = 2^{16}$ ).

1128 means of proximal terms or/and gradient correction terms at the (possible) cost of communication  
 1129 overhead [69] [56] [70] [127]. Or again, employing SGD optimizers, such as server-side momentum  
 1130 [47], and, more in general, adaptive gradient-based optimizers [101], i.e., incorporating adaptive  
 1131 learning rates, have been shown to mitigate the effect of heterogeneous data as well as reducing the  
 1132 total communication rounds to reach model convergence. Also experience-driven solutions have  
 1133 lately emerged to counterbalance non-IIDness and speed-up convergence; a deep reinforcement  
 1134 learning based mechanism that intelligently selects the participants for each FL round has been  
 1135 proposed in [124].

1136

1137 Authors of [145] experimentally show that test accuracy of FedAvg can be significantly increased  
 1138 in non-IID scenarios by providing a small subset of globally shared data (e.g., 5%); participants use  
 1139 their private dataset augmented with such data examples, provided by the server, to train their  
 1140 updates. Despite the effectiveness of the proposed solution, it has the cost of less decentralization  
 1141 and requires communicating the globally shared data to the participants. Authors also propose an  
 1142 alternative initialization of the global model; instead of a random initialization, the server trains  
 1143 a warm-up model using the shared data before broadcasting the model at the beginning of the  
 1144 learning task.

1145

1146 Authors of [131] observe two critical aspects of FedAvg, especially when dealing with non-  
 1147 IIDness. In fact, they argue that the additional on-device iterations between global synchronizations  
 1148 produce gradient biases, and that selecting a fraction of participants in each round results in an  
 1149 inconsistency between the optimization objectives and the real target distribution (the global model  
 1150 is trained by minimizing the empirical loss on data distributions that are, in general, different in  
 1151 each round of FedAvg). Since allowing multiple local iterations and selecting a part of clients are  
 1152 fundamental for the communication efficiency of FedAvg and its suitability in federated settings,  
 1153 authors of [131] propose two (distinct but jointly usable) strategies to alleviate such issues. They  
 1154 propose an Unbiased Gradient Aggregation (UGA) that performs what they call *keep-trace gradient*  
 1155 *descent optimization* for the first  $E - 1$  epochs, and then uses the whole data set to evaluate gradients  
 1156 during the last epoch. The key idea of *keep-trace gradient descent optimization* is preserving the  
 1157 functional relation, between  $w_t^{k(i)}$  and  $w_t^{k(i-1)}$  in round  $t$  for subsequent on-device iterations  $i$   
 1158 on client  $k$  (as usual,  $w$  indicates local/global model parameters) instead of passing for numerical  
 1159 values of gradients  $g_t^{k(i)}$ , such that in the last epoch they can calculate the gradient,  $g_t^k$ , against  
 1160  $w_t$  directly (considering the entire participant's data set). It is worth noting that, in UGA, the  
 1161 server gathers and aggregates thus calculated gradients  $g_t^k$  to produce the global model for the next  
 1162 iteration. On the other hand, to address the lack of a clear objective among subsequent rounds with  
 1163 different participants, authors propose FedMeta. The optimization process becomes a two-stage  
 1164 optimization: after each global aggregation (either performed following the baseline FedAvg or  
 1165 UGA), the server runs an additional gradient descent step using a special dataset,  $\mathcal{D}_{meta}$ . The  
 1166 rationale is that using such meta training set at server-side provides a clear and consistent objective  
 1167 in the learning process. Obviously, the composition of  $\mathcal{D}_{meta}$  is critical.

1168

1169 Authors of FedProx [69] tackle the potential model drift caused by non-IIDness by adding a  
 1170 proximal term to the local objective function instead of just heuristically tuning the number of local  
 1171 epochs; intuitively, the impact of local data is limited by restricting the locally-computed updates  
 1172 to be close to the current global model. Furthermore, FedProx allows for local solvers of choice, not  
 1173 limiting them to be SGD as happen for the traditional FedAvg. It is worth noting that FedProx is a  
 1174 generalization of FedAvg; if the multiplicative (hyper)parameter,  $\mu$ , that rules the proximal term  
 1175 in FedProx is set to 0 and the local solver of participants is restricted to be SGD, FedProx exactly  
 1176 matches FedAvg.

1177

1178

1177 Authors of SCAFFOLD [56] address the issue of drifting clients using control variates in FedAvg.  
1178 The idea is to align client updates by applying a correction term to the local gradients on each local  
1179 step. Each client computes its local control variate that represents the expected direction of the  
1180 local update while a global control variate that represents the aggregated direction in which the  
1181 server updates the global model is defined to be the uniform average of local control variates. Each  
1182 participant corrects its update by adding to the locally computed stochastic gradient the difference  
1183 between the global and the local control variate. The hypothetical case that motivates this strategy  
1184 is to have all clients computing the same update for the global model hence eliminating the model  
1185 drift. However, to achieve this, clients should communicate with each other every (either directly  
1186 or via parameter server) local gradient step, e.g. each client communicating its locally computed  
1187 gradient, that is unfeasible. Therefore, the local control variates and consequently the global control  
1188 variates are estimated throughout the process, and the global control variate is broadcasted to the  
1189 participants together with the model parameters at the beginning of every round by the server.

1190 FedDANE [70], inspired by DANE [110] and its inexact variant [102], combines the use of the  
1191 proximal term exploited in FedProx with a gradient correction term similarly to SCAFFOLD. The  
1192 update phase is a two-step process: to compute the gradient correction term and to inexactly solve  
1193 the Newton-type sub-problem, the locally computed gradients of the local objective functions  
1194 should be firstly collected and then averaged to approximate the full gradients. However, given  
1195 the realistic connection bottleneck in cross-device federated settings, it is unfeasible to gather all  
1196 the locally computed gradients; in FedDANE, the full gradients are approximated aggregating the  
1197 gradients of a randomly sub-sampled set of participants. It is worth noting that each update requires  
1198 two rounds of communication differently from the baseline FedAvg, FedProx and SCAFFOLD –  
1199 even though SCAFFOLD has to communicate in each round both the model parameters and the  
1200 control variates. Despite the theoretical convergence guarantee, FedDANE shows “disappointing  
1201 performance” in experimental evaluation compared to FedAvg and FedProx leaving doubts on the  
1202 robustness of theoretical assumptions.

1203 Authors of [101] propose an approach to decouple server and client learning rate and to exploit  
1204 adaptive learning rates on both client and server, with the primary objective of tackling client drift.  
1205 The idea is to have clients that leverage some *client optimizer* to minimize the loss on their local  
1206 dataset, while the server exploits a gradient-based *server optimizer* to minimize the loss across  
1207 participants. Building upon such general framework, namely FedOpt, they introduce and evaluate  
1208 some per-coordinate adaptive methods as server optimizers with SGD as client optimizer. In  
1209 practice, they implement three adaptive server optimizers, i.e. FedAdaGrad, FedYogi, and FedAdam  
1210 respectively being the federated versions of the well-known AdaGrad [83] [27], Yogi [136], and  
1211 ADAM. In their comparison with FedAvg<sup>18</sup> they also include FedAvgM [47]. They show that such  
1212 approaches are effective, in some circumstances “dramatically” effective with respect to FedAvg, in  
1213 mitigating client drift and, as a natural consequence, in reducing the total number of communication  
1214 rounds required for model convergence. Authors of [101] also provide theoretical convergence  
1215 analysis, and observe the need for a decaying learning rate at client-side.

1216

#### 1217 4.5 Handling Device Heterogeneity

1218 Device heterogeneity, i.e. device with diverse hardware characteristics or/and with different connec-  
1219 tivity (in general referred as *resources*), is common in cross-device federated settings. Such hetero-  
1220 geneity negatively influences the training process; for example, in federated learning frameworks  
1221

1222

1222 <sup>18</sup>It is worth noting that, under the proposed framework, FedAvg and FedAvgM [47], i.e. FedAvg with server-side momentum,  
1223 become specializations of the FedOpt family; the former uses SGD as both client and server optimizer with server learning  
1224 rate equal to 1, while the latter employs SGD with momentum as server optimizer.

1225



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<sup>19</sup> 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

<sup>19</sup>Authors of [125] consider a general definition of “resources” including, e.g., bandwidth, energy, time and monetary cost.

1275 being  $\tau$ , the number of local updates between two global synchronizations, and considering the  
 1276 resulting number of global synchronizations  $K$ , i.e.  $K = T/\tau$ , the total amount of consumed type- $m$   
 1277 resource is  $(T + 1)c_m + (K + 1)b_m$ , noting that the additional “+1” accounts for computing the last  
 1278 loss value after the last synchronization  $K$ . The objective is to minimize the global loss function by  
 1279 tuning  $\tau$  and  $K$  (and, consequently,  $T$ ) such that the total amount of consumed type- $m$  resource is  
 1280 not greater than the resource budget  $R_m$  (each type- $m$  resource has a certain budget associated).  
 1281 Such minimization problem is approximately solved by leveraging a theoretical convergence upper  
 1282 bound of the canonical distributed gradient descent after  $T$  iterations, although assuming that  
 1283 the loss function is (i) convex, (ii)  $\rho$ -Lipschitz and (iii)  $\beta$ -smooth. In the convergence analysis,  
 1284 authors also define an upper bound for gradient divergence, i.e. an upper bound of the divergence  
 1285 between the gradient of the local loss function and the gradient of the global loss function, that  
 1286 depends on how the data is distributed among different participants, hence taking into account the  
 1287 non-IIDness of data. We redirect to the full paper for the complete theoretical analysis. In a nutshell,  
 1288 the proposed control algorithm recomputes the optimal<sup>20</sup>  $\tau$ , hereinafter referred as  $\tau^*$ , during each  
 1289 aggregation step via linear search on integer values of  $\tau$  accordingly to the most updated parameter  
 1290 estimations needed to approximately solves the minimization problem mentioned above.

1291 In regards to peer-to-peer frameworks, BACombo (already presented in 3.3.2) interestingly  
 1292 leverages a bandwidth-aware worker selection, i.e the peers to be requested for model segments are  
 1293 not trivially chosen randomly. To reduce transmission time, peers with faster network connections  
 1294 should be preferred. However, it is not easy to know the network condition of a certain peer a priori.  
 1295 The proposed solution exploit a multi-armed bandit algorithm [5]; each participant, with probability  
 1296  $\epsilon$ , either explores the network conditions of peers by selecting them randomly or exploits its already  
 1297 acquired knowledge — each participant maintains a table, that is updated each time a peer is picked  
 1298 for communication, that contains historical indications about the network state of that peer — by  
 1299 greedily selecting the peers with best network conditions.

1300

#### 1301 4.6 Defending against Poisoning

1302 From being passive data providers, in cloud-based ML, participants become active entities in the  
 1303 learning process of decentralized training: they locally compute updates and observe intermediate  
 1304 model states. Although this design is the cornerstone to improve several aspects of traditional  
 1305 ML/DL, it exposes the system to a larger variety of attacks from malicious learners, since partici-  
 1306 pants, in theory, can contribute with arbitrary updates, and could try to manipulate the learning  
 1307 process for diverse scopes (e.g., merely hampering the convergence, forcing other participants to  
 1308 over-expose their contribution or backdooring the system), while making their detection harder  
 1309 since the raw data are not accessible. This is known as model poisoning, besides the more traditional  
 1310 data poisoning. We redirect the reader to [79] for a complete understanding of the threat model and  
 1311 of the attack variety. We present here some strategies to detect and/or neutralize poisoning attacks.

1312

1313 Authors of [128] (SLSGD) propose a variation of FedAvg to address non-IIDness and to tolerate  
 1314 data poisoning attacks (evaluated by simulating the attack through label flipping). They act on the  
 1315 baseline FedAvg algorithm by varying (i) the aggregation step and (ii) the new-model generation  
 1316 step; (i) instead of aggregating the updates by averaging, they use a trimmed mean to (try to) filter  
 1317 out poisoned updates, and (ii) instead of replacing the previous global model with the resulting  
 1318 aggregated model, they use a moving average between the previous and the just aggregated model  
 1319

1320

1320 <sup>20</sup>It is worth noting that, intuitively, if the resource budget is unlimited,  $\tau^*$  is equal to 1, i.e. global synchronization after each  
 1321 local update, while in presence of budget constraints it may be convenient investing the resource for local computations  
 1322 rarefying the global synchronizations, i.e.  $\tau^* > 1$ .

1323



1324 to limit the influence of non-IID datasets and to mitigate the extra variance caused by such “robust”  
 1325 aggregation.

1326 In [33], authors propose a defense against sybil-based poisoning (precisely, label-flipping and  
 1327 backdoor poisoning), namely FoolsGold, targeting a federated learning framework where par-  
 1328 ticipants upload locally computed gradients to the (honest) aggregator. The idea is to identify  
 1329 malicious colluding participants, i.e. poisoning sybils, by monitoring the diversity of participants’  
 1330 update; sybils are supposed to share a common objective and the directions of poisoning gradients  
 1331 should seem unusually similar respect to updates from honest learners. In a nutshell, FoolsGold  
 1332 maintains an historical aggregate of updates per participant at server side, i.e. the cumulative  
 1333 sums of its updates so far, and it measures the cosine similarity between couple of participants’  
 1334 historical aggregates before each aggregation step — the rationale behind this strategy is that gra-  
 1335 dients resulting from single local iteration of SGD can be very similar in directions even among  
 1336 honest clients, however colluding parties will share the same objective in the long run, limiting the  
 1337 effectiveness of poisoning throughout the training process by accordingly re-scaling the learning  
 1338 rate of participants that are deemed as possible sybils. The clear limit of FoolsGold — apart from  
 1339 being incompatible with secure aggregation and assuming honest aggregator — is that it is designed  
 1340 to look for sybils, hence a single participant adversary can remain undetected.

1341 Authors of [141] propose a defense against poisoning, specifically targeting label flipping and  
 1342 semantic backdoor attacks, in a synchronous federated learning framework accounting also for  
 1343 non-IIDness. Differently from FoolsGold [33], their strategy actively leverages on clients; the server  
 1344 asks to the participants to evaluate some sub-models, each one derived from the aggregation of  
 1345 disjoint subsets of the model updates related to a certain round, and they provide back to the  
 1346 server an indication about the correctness in the classification task of such sub-models, tested  
 1347 on their private dataset, in the form of a binary matrix (obviously, a certain participant cannot  
 1348 receive a sub-model derived from its own contribution). Thanks to the gathered matrices, the server  
 1349 computes a penalizing coefficient for each sub-update to weigh the aggregation of such sub-models  
 1350 (for example, if more than half of the clients report the anomaly for the same sub-model, it should  
 1351 be zero-weighted). Authors highlight that their solution can be also combined to FoolsGold [33], e.g.  
 1352 to detect single-participant attack.

1353 Similarly to [33] and [141], authors of [66] use a server-side pre-trained autoencoder model to  
 1354 detect abnormal weight updates that are then accordingly penalized during the aggregation.

1355

## 1356 5 OPEN PROBLEMS AND FUTURE DIRECTIONS

1357 As an obvious observation, we remark that data-sequential approaches are only limited to Cross-silo  
 1358 federated settings, where the number of participants is limited (see Table 1). At the same time, (data-  
 1359 parallel) star-shaped synchronous systems and related improvements (i.e., 44 out of 53 surveyed  
 1360 solutions) have dominated the early years of decentralized learning, pushed by the Google’s FedAvg  
 1361 baseline and, not surprisingly, the first real-world large-scale decentralized learning system for  
 1362 Cross-device federated settings has followed this trend [12]. Nevertheless, we stress the evidence  
 1363 that relaxing the synchronous constraint for aggregating updates in star-shaped systems mitigates  
 1364 the struggles in handling a large amount of heterogeneous devices, while introducing degrees of  
 1365 uncertainty that hamper the theoretical comprehension of the system’s behaviour in real scenarios  
 1366 (e.g. FedAsync solution adopts this strategy). At the other end of the spectrum, we observe a reduced  
 1367 portion of fully decentralized solutions (only 5 systems out of 53, with one of them, i.e. SAPS-  
 1368 PSGD [117], that leverages a central entity for coordination). In addition, the MEC-architecture  
 1369 has demonstrated to effectively help in scaling the learning process and is increasingly adopted; in  
 1370 Table 1, we report 3 works explicitly considering this architecture. Indeed, that allows to favour  
 1371 the exploration and ease the implementation of hierarchical solutions, such as star-shaped both  
 1372

1373 between devices and edge servers, and between edge servers and the cloud. To conclude, in the  
1374 next subsections, we will present other open challenges that will likely influence the incoming  
1375 future of decentralized learning systems, by also sketching possible and most promising directions  
1376 for future research.

## 1377 5.1 Rethinking the Traditional ML Workflow for Federated Learning

1378 The literature explored in this survey proposes solutions to the main challenges of employing  
1379 federated learning systems in real-world scenarios. However, most works suppose that the hyper-  
1380 parameters (e.g., the neural network's architecture, regularization techniques, and optimizers) of  
1381 the model to be trained have been already established, and typically the focus is not about the  
1382 tuning of their determination. Furthermore, decentralized learning systems introduce additional  
1383 algorithm-specific hyperparameters (e.g., the number of local epochs or the number of participants  
1384 involved per round) that significantly influence the performance of the adopted solution. While in  
1385 cloud-centric DL it is feasible to run many rounds of training to empirically search the hyperparam-  
1386 eters space towards optimality, this strategy is probably infeasible for cross-silo settings and surely  
1387 incompatible with cross-device settings. Hence, we expect that hyperparameter optimization that  
1388 targets the communication and computation overhead on the devices that compose the federation,  
1389 and not only aiming at the best accuracy of models as happens in datacenter optimizations, will  
1390 gain traction, by fostering the development of easy-to-tune and/or auto-tuning algorithms for  
1391 federated settings (e.g., [14] – explored in Section 4 – and [41]).

1392 Another relevant phase of the traditional workflow in cloud-centric ML, which is reshaped by the  
1393 design of decentralized learning systems, relates to the debugging of trained models' behaviour. In  
1394 fact, preventing the access to the raw data by design does preclude modelers and practitioners from  
1395 directly investigating the causes of the detected problems (e.g., investigating missclassification,  
1396 noticing evident bias in the training set, identifying outliers, manually adding or adjusting labels),  
1397 i.e. manual data inspection is impossible [6]. Connected to that, the design and implementation  
1398 of privacy-preserving techniques to enable the debug phase also for federated learning systems  
1399 are open areas of research. For example, in [6], the privacy concerns are overtaken by using  
1400 privacy-preserving Generative Adversarial Network trained in a federated fashion, thus enabling  
1401 the debug on synthetic data examples that conjugate the trade-off between information leakage  
1402 and debugging utility.

## 1403 5.2 Designing Incentive Mechanisms

1404 Another assumption typically made in the FL-related literature is that the (selected) learners are  
1405 willing to participate. Leaving aside for a moment the privacy concerns that may discourage  
1406 participants, another factor that can determine the reluctance in being involved in federated  
1407 learning processes is the associated overhead, in terms of computation and communication. Self-  
1408 interested mobile devices may be unwilling to cooperate without receiving adequate rewards  
1409 [55]. Such considerations may be exacerbated in cross-silo federated settings, where competitors  
1410 should collaborate for a common objective, while they may have local data different in quality  
1411 (i.e., an organization with rich and high-quality local data would not be willing to participate  
1412 in a federated learning process and sharing, for free, the acquired final knowledge with other  
1413 competitors that have contributed much less in the learned model due to scarce-quality data).  
1414 Furthermore, the revenue generated from the built model will come only afterwards [133]. In this  
1415 direction, solutions to properly reward participants and attracting data owners with high-quality  
1416 data, e.g. more conspicuous rewards for participants with higher quality of local data, are emerging  
1417 (e.g., [55], [133]). Designing effective incentive mechanisms will be fundamental for the spreading  
1418 of decentralized learning in real-world scenarios.

1420  
1421

### 1422 5.3 Towards Model Heterogeneity and Personalization

1423 As we have seen, in federated settings, different kinds of heterogeneity must be addressed, from  
1424 system heterogeneity (i.e., device with different resource budgets) to data heterogeneity (i.e.,  
1425 non-IIDness). We highlight an additional facet of heterogeneity that regards the local model  
1426 architecture: each participant of the learning process can design its own model accordingly to its  
1427 needs. This degree of freedom would further favour the collaboration among institutions – under  
1428 the perspective of intellectual property related to the tailored model architecture – and can be  
1429 also leveraged to favour the inclusion of more resource-constrained edge devices in the learning  
1430 process. Transfer learning and knowledge distillation are investigated to effectively enabling such  
1431 independence improvements among participants (e.g., [65]). Besides model heterogeneity, model  
1432 personalization, i.e. fitting the global model to the participant-specific local data, would represent  
1433 an additional tool to tackle non-IIDness [62].  
1434

### 1435 5.4 Going beyond Supervised Learning

1436 It is important to underline once more that almost all the cited works in this survey suppose labeled  
1437 data examples within supervised learning contexts. However, in real federated settings it could not  
1438 be straightforward to automatically or to manually label data samples; while systems to favour the  
1439 collection of user-annotated examples are arising (e.g., [78]), the huge amount of unlabeled raw  
1440 data, that will be produced in the next years at the edge of the network, may not be adequately  
1441 exploited by only supervised learning techniques. Anyway, opening up to semi-supervised [52],  
1442 unsupervised or to reinforcement learning approaches would require similar issues in terms of  
1443 privacy guarantees, heterogeneity, communication efficiency and scalability.  
1444

### 1445 5.5 User Perception of FL Privacy Guarantees

1446 The rising regulations about privacy protection would ideally require the express consent of users  
1447 for sensitive-data collection and processing. Decentralized learning techniques naturally shape  
1448 the principles of focused data collection and minimization, on which most of the privacy-related  
1449 regulations build on as well. However, we might wonder if the average user fully understands  
1450 the privacy benefits and limitations that come with the design of decentralized learning systems,  
1451 and in particular with privacy-preserving decentralized learning systems (e.g., differential private  
1452 decentralized training). In fact, only if the user is aware of the guarantees about privacy protection,  
1453 she or he can consciously decide whether and which data letting be involved in possible decentral-  
1454 ized learning processes. Moreover, different users may value privacy aspects differently, eventually  
1455 entailing fine granular and user-specific tuning of privacy guarantees, an aspect that has not been  
1456 thoroughly explored yet. Orthogonally, there is no clear consensus on how to choose privacy  
1457 parameters (e.g.,  $\epsilon$  for  $\epsilon$ -DP mechanisms) [28]. Fostering and creating a shared consensus about the  
1458 adequate level of privacy in collaborative learning systems is another key aspect for the incoming  
1459 future, as well as fully understanding and addressing the specific privacy preferences of educated  
1460 users (i.e., users who have full comprehension of the implications of the privacy technology used).  
1461

### 1462 5.6 Fairness and Sources of Bias in Decentralized Learning

1463 The relevant objective of ensuring fairness does not strictly relate to decentralized learning; it  
1464 is a recognized and well-known issue in traditional ML/DL. However, some unique and peculiar  
1465 traits of decentralized learning systems open up to new directions for future research. In fact,  
1466 especially in cross-device settings, practical assumptions and requirements about the (selected)  
1467 per-round participants can generate bias in the training data, which in turn might make the model  
1468 unfair, e.g., under-represented groups in training samples may receive lower-quality predictions, or  
1469  
1470

1471 individuals that should be treated similarly by the model receive significantly different outcomes,  
1472 or again the trained model might show prejudices against some sensitive subgroups of individuals.  
1473 By going into practical details and consequences, for example, the proposed implementation of FL  
1474 for Android mobile devices includes in the training rounds only the devices that are (i) connected  
1475 to unmetered network, (ii) charging, and (iii) that respond within a time-out (also the involved  
1476 devices have to meet some hardware requirements, i.e., memory); this may lead to sample a  
1477 biased population of participants. Solutions for more flexible device participation (e.g., [105]) can  
1478 mitigate such phenomenon. Similar observations raise from other strategies such as prioritizing  
1479 fast connected devices (e.g., in [117] or [50]). Furthermore, also imbalanced data among nodes  
1480 can represent a source of bias [26], and this has demonstrated to be more typical of cross-device  
1481 settings. Another factor that makes fairness challenging in decentralized learning systems lies in  
1482 the privacy-preserving design of such approaches: usually data are not directly accessible to search  
1483 for bias in data samples.

1484

1485

### 5.7 Towards Fully Decentralized Systems at Scale

1486

1487

1488

1489

1490

1491

1492

1493

1494

1495

1496

1497

1498

1499

1500

1501

1502

1503

1504

1505

1506

1507

1508

## 6 CONCLUDING REMARKS

1509

1510

1511

1512

1513

1514

1515

1516

1517

1518

1519

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

<sup>21</sup>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 Konečný, 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 Konečný, 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.



- 1569 [22] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew  
1570 Senior, Paul Tucker, Ke Yang, et al. 2012. Large scale distributed deep networks. In *Advances in neural information*  
1571 *processing systems*. 1223–1231.
- 1572 [23] Whitfield Diffie and Martin Hellman. 1976. New directions in cryptography. *IEEE transactions on Information Theory*  
22, 6 (1976), 644–654.
- 1573 [24] Tung V. Doan, Zhongyi Fan, Giang T. Nguyen, Hani Salah, Dongho You, and Frank HP Fitzek. 2020. Follow Me, If You  
1574 Can: A Framework for Seamless Migration in Mobile Edge Cloud. *IEEE INFOCOM Workshops (2020)*, 1178–11183.
- 1575 [25] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In  
1576 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4829–4837.
- 1577 [26] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. 2020. Self-Balancing Federated  
1578 Learning With Global Imbalanced Data in Mobile Systems. *IEEE Transactions on Parallel and Distributed Systems* 32, 1  
(2020), 59–71.
- 1579 [27] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic  
1580 optimization. *Journal of machine learning research* 12, 7 (2011).
- 1581 [28] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. 2019. Differential Privacy in Practice: Expose your Epsilons!  
1582 *Journal of Privacy and Confidentiality* 9, 2 (2019).
- 1583 [29] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends®*  
1584 *in Theoretical Computer Science* 9, 3–4 (2014), 211–407.
- 1585 [30] EU. [n.d.]. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. URL  
1586 <https://eur-lex.europa.eu/legal-content/EN/TXT/>.
- 1587 [31] Romano Fantacci and Benedetta Picano. 2020. Federated learning framework for mobile edge computing networks.  
1588 *CAA Transactions on Intelligence Technology* 5, 1 (2020), 15–21.
- 1589 [32] Yingwei Fu, Huaimin Wang, Kele Xu, Haibo Mi, and Yijie Wang. 2019. Mixup Based Privacy Preserving Mixed  
1590 Collaboration Learning. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE,  
1591 275–2755.
- 1592 [33] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2018. Mitigating sybils in federated learning poisoning. *arXiv*  
1593 *preprint arXiv:1808.04866* (2018).
- 1594 [34] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective.  
1595 *arXiv preprint arXiv:1712.07557* (2017).
- 1596 [35] Lodovico Giaretta and Šarūnas Girdzijauskas. 2019. Gossip learning: Off the beaten path. In *2019 IEEE International*  
1597 *Conference on Big Data (Big Data)*. IEEE, 1117–1124.
- 1598 [36] Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of*  
1599 *Network and Computer Applications* 116 (2018), 1–8.
- 1600 [37] Meng Hao, Hongwei Li, Guowen Xu, Sen Liu, and Haomiao Yang. 2019. Towards Efficient and Privacy-Preserving  
1601 Federated Deep Learning. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.
- 1602 [38] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon,  
1603 and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).
- 1604 [39] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. 2018. Gossiping GANs. In *Proceedings of the Second Workshop*  
1605 *on Distributed Infrastructures for Deep Learning: DIDL*, Vol. 22.
- 1606 [40] Valentin Hartmann and Robert West. 2019. Privacy-Preserving Distributed Learning with Secret Gradient Descent.  
1607 *arXiv preprint arXiv:1906.11993* (2019).
- 1608 [41] Chaoyang He, Murali Annavam, and Salman Avestimehr. 2020. FedNAS: Federated Deep Learning via Neural  
1609 Architecture Search. *arXiv preprint arXiv:2004.08546* (2020).
- 1610 [42] Chaoyang He, Conghui Tan, Hanlin Tang, Shuang Qiu, and Ji Liu. 2019. Central server free federated learning over  
1611 single-sided trust social networks. *arXiv preprint arXiv:1910.04956* (2019).
- 1612 [43] István Hegedűs, Gábor Danner, and Márk Jelasity. 2019. Gossip Learning as a Decentralized Alternative to Federated  
1613 Learning. In *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 74–90.
- 1614 [44] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint*  
1615 *arXiv:1503.02531* (2015).
- 1616 [45] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage  
1617 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).

- 1618 [48] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. 2018. LoAdaBoost: Loss-Based AdaBoost  
 1619 Federated Machine Learning on medical Data. *arXiv preprint arXiv:1811.12629* (2018).
- 1620 [49] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. 2018. Communication-  
 1621 efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv*  
 1622 *preprint arXiv:1811.11479* (2018).
- 1623 [50] Jingyan Jiang, Liang Hu, Chenghao Hu, Jiatae Liu, and Zhi Wang. 2020. BACoMbo—Bandwidth-Aware Decentralized  
 1624 Federated Learning. *Electronics* 9, 3 (2020), 440.
- 1625 [51] Richeng Jin, Yufan Huang, Xiaofan He, Huaiyu Dai, and Tianfu Wu. 2020. Stochastic-Sign SGD for Federated Learning  
 1626 with Theoretical Guarantees. *arXiv preprint arXiv:2002.10940* (2020).
- 1627 [52] Yilun Jin, Xiguang Wei, Yang Liu, and Qiang Yang. 2020. A Survey towards Federated Semi-supervised Learning.  
 1628 *arXiv preprint arXiv:2002.11545* (2020).
- 1629 [53] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith  
 1630 Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in  
 1631 federated learning. *arXiv preprint arXiv:1912.04977* (2019).
- 1632 [54] Michael Kamp, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel. 2018.  
 1633 Efficient decentralized deep learning by dynamic model averaging. In *Joint European Conference on Machine Learning*  
 1634 *and Knowledge Discovery in Databases*. Springer, 393–409.
- 1635 [55] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. 2019. Incentive mechanism for reliable  
 1636 federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of*  
 1637 *Things Journal* 6, 6 (2019), 10700–10714.
- 1638 [56] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha  
 1639 Suresh. 2019. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. *arXiv preprint*  
 1640 *arXiv:1910.06378* (2019).
- 1641 [57] Kimon Karras, Evangelos Pallas, George Mastorakis, Yannis Nikoloudakis, Jordi Mongay Batalla, Constandinos X.  
 1642 Mavroumoustakis, and Evangelos K. Markakis. 2020. A Hardware Acceleration Platform for AI-Based Inference at the  
 1643 Edge. *Circuits Syst. Signal Process.* 39, 2 (2020), 1059–1070.
- 1644 [58] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. 2018. On-device federated learning via blockchain  
 1645 and its latency analysis. *arXiv preprint arXiv:1808.03949* (2018).
- 1646 [59] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*  
 1647 (2014).
- 1648 [60] Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed  
 1649 machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).
- 1650 [61] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016.  
 1651 Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).
- 1652 [62] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. 2020. Survey of Personalization Techniques for Federated  
 1653 Learning. *arXiv preprint arXiv:2003.08673* (2020).
- 1654 [63] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.
- 1655 [64] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. 2019. Federated learning  
 1656 for keyword spotting. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing*  
 1657 *(ICASSP)*. IEEE, 6341–6345.
- 1658 [65] Daliang Li and Junpu Wang. 2019. FedMD: Heterogenous Federated Learning via Model Distillation. *arXiv preprint*  
 1659 *arXiv:1910.03581* (2019).
- 1660 [66] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen. 2019. Abnormal client behavior detection in federated  
 1661 learning. *arXiv preprint arXiv:1910.09933* (2019).
- 1662 [67] Tian Li, Zaoxing Liu, Vyas Sekar, and Virginia Smith. 2019. Privacy for Free: Communication-Efficient Learning with  
 1663 Differential Privacy Using Sketches. *arXiv preprint arXiv:1911.00972* (2019).
- 1664 [68] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and  
 1665 Future Directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.
- 1666 [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.



- 1667 [73] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit  
1668 Niyato, and Chunyan Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE*  
1669 *Communications Surveys & Tutorials* (2020).
- 1670 [74] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. 2017. Deep gradient compression: Reducing the  
1671 communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887* (2017).
- 1672 [75] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. 2019. Edge-Assisted Hierarchical Federated Learning with  
1673 Non-IID Data. *arXiv preprint arXiv:1905.06641* (2019).
- 1674 [76] Menghan Liu, Haotian Jiang, Jia Chen, Alaa Badokhon, Xuetao Wei, and Ming-Chun Huang. 2016. A collaborative  
1675 privacy-preserving deep learning system in distributed mobile environment. In *2016 International Conference on*  
1676 *Computational Science and Computational Intelligence (CSCI)*. IEEE, 192–197.
- 1677 [77] Wei Liu, Li Chen, Yunfei Chen, and Wenyi Zhang. 2020. Accelerating Federated Learning via Momentum Gradient  
1678 Descent. *IEEE Transactions on Parallel and Distributed Systems* 31, 8 (2020), 1754–1766.
- 1679 [78] Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and  
1680 Qiang Yang. 2020. FedVision: An Online Visual Object Detection Platform Powered by Federated Learning. *arXiv*  
1681 *preprint arXiv:2001.06202* (2020).
- 1682 [79] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to Federated Learning: A Survey. *arXiv preprint arXiv:2003.02133*  
1683 (2020).
- 1684 [80] Evangelos K. Markakis, Kimon Karras, Nikolaos Zotos, Anargyros Sideris, Theoharris Moysiadis, Angelo Corsaro,  
1685 George Alexiou, Charalabos Skianis, George Mastorakis, Constandinos X. Mavromoustakis, and Evangelos Pallis.  
1686 2017. EXEGESIS: Extreme Edge Resource Harvesting for a Virtualized Fog Environment. *IEEE Commun. Mag.* 55, 7  
1687 (2017), 173–179.
- 1688 [81] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. 2016. Communication-efficient learning of  
1689 deep networks from decentralized data. *arXiv preprint arXiv:1602.05629* (2016).
- 1690 [82] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private language  
1691 models without losing accuracy. *arXiv preprint arXiv:1710.06963* (2017).
- 1692 [83] H Brendan McMahan and Matthew Streeter. 2010. Adaptive bound optimization for online convex optimization.  
1693 *arXiv preprint arXiv:1002.4908* (2010).
- 1694 [84] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature  
1695 leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 691–706.
- 1696 [85] Jed Mills, Jia Hu, and Geyong Min. 2019. Communication-Efficient Federated Learning for Wireless Edge Intelligence  
1697 in IoT. *IEEE Internet of Things Journal* (2019).
- 1698 [86] Akinori Mitani, Abigail Huang, Subhashini Venugopalan, Greg S Corrado, Lily Peng, Dale R Webster, Naama Hammel,  
1699 Yun Liu, and Avinash V Varadarajan. 2020. Author Correction: Detection of anaemia from retinal fundus images via  
1700 deep learning. *Nature Biomedical Engineering* 4, 2 (2020), 242–242.
- 1701 [87] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings*  
1702 *of the 27th international conference on machine learning (ICML-10)*, 807–814.
- 1703 [88] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust de-anonymization of large datasets (how to break anonymity  
1704 of the Netflix prize dataset). *University of Texas at Austin* (2008).
- 1705 [89] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Comprehensive privacy analysis of deep learning: Stand-alone  
1706 and federated learning under passive and active white-box inference attacks. *arXiv preprint arXiv:1812.00910* (2018).
- 1707 [90] Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed. 2020. Federated Learning for Wireless Communications:  
1708 Motivation, Opportunities, and Challenges. *IEEE Communications Magazine* 58, 6 (2020), 46–51.
- 1709 [91] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in  
1710 mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.
- 1711 [92] Kenta Niwa, Noboru Harada, Guoqiang Zhang, and W Bastiaan Kleijn. 2020. Edge-consensus Learning: Deep Learning  
1712 on P2P Networks with Nonhomogeneous Data. In *Proceedings of the 26th ACM SIGKDD International Conference on*  
1713 *Knowledge Discovery & Data Mining*. 668–678.
- 1714 [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.
- 1715 [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),

- 1716 1333–1345.
- 1717 [98] Tran Thi Phuong et al. 2019. Privacy-preserving deep learning via weight transmission. *IEEE Transactions on*
- 1718 *Information Forensics and Security* 14, 11 (2019), 3003–3015.
- 1719 [99] Maarten G Poitrot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar.
- 1720 2019. Split Learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115* (2019).
- 1721 [100] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated Learning for Emoji
- 1722 Prediction in a Mobile Keyboard. *arXiv preprint arXiv:1906.04329* (2019).
- 1723 [101] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and
- 1724 H Brendan McMahan. 2020. Adaptive Federated Optimization. *arXiv preprint arXiv:2003.00295* (2020).
- 1725 [102] Sashank J Reddi, Jakub Konečný, Peter Richtárik, Barnabás Póczós, and Alex Smola. 2016. Aide: Fast and communica-
- 1726 tion efficient distributed optimization. *arXiv preprint arXiv:1608.06879* (2016).
- 1727 [103] Amirhossein Reiszadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. 2019. Fedpaq:
- 1728 A communication-efficient federated learning method with periodic averaging and quantization. *arXiv preprint*
- 1729 *arXiv:1909.13014* (2019).
- 1730 [104] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. 2019. Baintorrent: A
- 1731 peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731* (2019).
- 1732 [105] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong. 2020. Towards Flexible Device Participation in
- 1733 Federated Learning for Non-IID Data. *arXiv preprint arXiv:2006.06954* (2020).
- 1734 [106] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Robust and communication-efficient
- 1735 federated learning from non-iid data. *IEEE transactions on neural networks and learning systems* (2019).
- 1736 [107] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Sparse binary compression:
- 1737 Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural*
- 1738 *Networks (IJCNN)*. IEEE, 1–8.
- 1739 [108] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. 2020. Federated Learning with Cooperating Devices: A Consensus
- 1740 Approach for Massive IoT Networks. *IEEE Internet of Things Journal* (2020).
- 1741 [109] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
- 1742 [110] Ohad Shamir, Nati Srebro, and Tong Zhang. 2014. Communication-efficient distributed optimization using an
- 1743 approximate newton-type method. In *International conference on machine learning*. 1000–1008.
- 1744 [111] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. 2019. Detailed comparison of communi-
- 1745 cation efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145* (2019).
- 1746 [112] Jinhyun So, Basak Guler, and A Salman Avestimehr. 2020. Turbo-Aggregate: Breaking the Quadratic Aggregation
- 1747 Barrier in Secure Federated Learning. *arXiv preprint arXiv:2002.04156* (2020).
- 1748 [113] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. 2018. Human Activity Recognition Using Federated
- 1749 Learning. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Com-*
- 1750 *munications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications*
- 1751 *(ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE, 1103–1111.
- 1752 [114] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout:
- 1753 a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014),
- 1754 1929–1958.
- 1755 [115] Nikko Strom. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual*
- 1756 *Conference of the International Speech Communication Association*.
- 1757 [116] Gábor J Székely, Maria L Rizzo, Nail K Bakirov, et al. 2007. Measuring and testing dependence by correlation of
- 1758 distances. *The annals of statistics* 35, 6 (2007), 2769–2794.
- 1759 [117] Zhenheng Tang, Shaohuai Shi, and Xiaowen Chu. 2020. Communication-efficient decentralized learning with
- 1760 sparsification and adaptive peer selection. *arXiv preprint arXiv:2002.09692* (2020).
- 1761 [118] Zeyi Tao and Qun Li. 2018. esgd: Communication efficient distributed deep learning on the edge. In *{USENIX}*
- 1762 *Workshop on Hot Topics in Edge Computing (HotEdge 18)*.
- 1763 [119] Chandra Thapa, MAP Chamikara, and Seyit Camtepe. 2020. SplitFed: When Federated Learning Meets Split Learning.
- 1764 *arXiv preprint arXiv:2004.12088* (2020).
- 1765 [120] Aleksei Triastcyn and Boi Faltings. 2019. Federated Learning with Bayesian Differential Privacy. *arXiv preprint*
- 1766 *arXiv:1911.10071* (2019).
- 1767 [121] Aleksei Triastcyn and Boi Faltings. 2019. Improved Accounting for Differentially Private Learning. *arXiv preprint*
- 1768 *arXiv:1901.09697* (2019).
- 1769 [122] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A
- 1770 hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial*
- 1771 *Intelligence and Security*. 1–11.

- 1765 [123] Praneeth Vepakomma, Otkrist Gupta, Abhimanyu Dubey, and Ramesh Raskar. 2019. Reducing leakage in distributed  
1766 deep learning for sensitive health data. *arXiv preprint arXiv:1812.00564* (2019).
- 1767 [124] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. 2020. Optimizing Federated Learning on Non-IID Data with  
1768 Reinforcement Learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 1698–1707.
- 1769 [125] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. 2019.  
1770 Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in  
1771 Communications* 37, 6 (2019), 1205–1221.
- 1772 [126] Wikipedia. [n.d.]. Facebook–Cambridge Analytica data scandal. URL [https://en.wikipedia.org/wiki/Facebook%27s\\_textdashCambridge\\_Analytica\\_data\\_scandal](https://en.wikipedia.org/wiki/Facebook%27s_textdashCambridge_Analytica_data_scandal). Accessed on May 2020.
- 1773 [127] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. Asynchronous federated optimization. *arXiv preprint  
1774 arXiv:1903.03934* (2019).
- 1775 [128] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2019. SLSGD: Secure and Efficient Distributed On-device Machine  
1776 Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.
- 1777 [129] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications.  
1778 *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.
- 1779 [130] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise  
1780 Beaufays. 2018. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint  
1781 arXiv:1812.02903* (2018).
- 1782 [131] Xin Yao, Tianchi Huang, Rui-Xiao Zhang, Ruiyu Li, and Lifeng Sun. 2019. Federated Learning with Unbiased Gradient  
1783 Aggregation and Controllable Meta Updating. *arXiv preprint arXiv:1910.08234* (2019).
- 1784 [132] Chun-Hsien Yu, Chun-Nan Chou, and Emily Chang. 2019. Distributed Layer-Partitioned Training for Privacy-  
1785 Preserved Deep Learning. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE,  
1786 343–346.
- 1787 [133] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. 2020. A  
1788 Fairness-aware Incentive Scheme for Federated Learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics,  
1789 and Society*. 393–399.
- 1790 [134] Qian Yu, Songze Li, Netanel Raviv, Seyed Mohammadreza Mousavi Kalan, Mahdi Soltanolkotabi, and Salman Aves-  
1791 timehr. 2019. Lagrange Coded Computing: Optimal Design for Resiliency, Security, and Privacy. In *International  
1792 Conference on Artificial Intelligence and Statistics (AISTATS 2019)*.
- 1793 [135] Zhengxin Yu, Jia Hu, Geyong Min, Haochuan Lu, Zhiwei Zhao, Haozhe Wang, and Nektarios Georgalas. 2018.  
1794 Federated learning based proactive content caching in edge computing. In *2018 IEEE Global Communications Conference  
1795 (GLOBECOM)*. IEEE, 1–6.
- 1796 [136] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. 2018. Adaptive methods for  
1797 nonconvex optimization. In *Advances in neural information processing systems*. 9793–9803.
- 1798 [137] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. 2019. Deep learning in mobile and wireless networking: A survey.  
1799 *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2224–2287.
- 1800 [138] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional mechanism: regression  
1801 analysis under differential privacy. *arXiv preprint arXiv:1208.0219* (2012).
- 1802 [139] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new  
1803 perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.
- 1804 [140] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. iDLG: Improved Deep Leakage from Gradients. *arXiv preprint  
1805 arXiv:2001.02610* (2020).
- 1806 [141] Lingchen Zhao, Shengshan Hu, Qian Wang, Jianlin Jiang, Chao Shen, and Xiangyang Luo. 2019. Shielding Collaborative  
1807 Learning: Mitigating Poisoning Attacks through Client-Side Detection. *arXiv preprint arXiv:1910.13111* (2019).
- 1808 [142] Lingchen Zhao, Qian Wang, Qin Zou, Yan Zhang, and Yanjiao Chen. 2019. Privacy-preserving collaborative deep  
1809 learning with unreliable participants. *IEEE Transactions on Information Forensics and Security* 15 (2019), 1486–1500.
- 1810 [143] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. 2019. Deep learning and its  
1811 applications to machine health monitoring. *Mechanical Systems and Signal Processing* 115 (2019), 213–237.
- 1812 [144] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, and Shui Yu. 2019. Multi-Task Network Anomaly Detection using Federated  
1813 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.

- 1814 [148] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Advances in Neural Information*  
1815 *Processing Systems*. 14747–14756.

1816

1817

1818

1819

1820

1821

1822

1823

1824

1825

1826

1827

1828

1829

1830

1831

1832

1833

1834

1835

1836

1837

1838

1839

1840

1841

1842

1843

1844

1845

1846

1847

1848

1849

1850

1851

1852

1853

1854

1855

1856

1857

1858

1859

1860

1861

1862