A Survey on Federated Learning for Resource-Constrained IoT Devices

Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, *Member, IEEE*, Jian Li, *Member, IEEE*, and M. Hadi Amini, *Member, IEEE*

***Abstract*—Federated learning (FL) is a distributed machine learning strategy that generates a global model by learning from multiple decentralized edge clients. FL enables on-device training, keeping the client's local data private, and further, updating the global model based on the local model updates. While FL methods offer several advantages, including scalability and data privacy, they assume there are available computational resources at each edge-device/client. However, the Internet-of-Things (IoT)-enabled devices, e.g., robots, drone swarms, and low-cost computing devices (e.g., Raspberry Pi), may have limited processing ability, low bandwidth and power, or limited storage capacity. In this survey article, we propose to answer this question: how to train distributed machine learning models for resource-constrained IoT devices? To this end, we first explore the existing studies on FL, relative assumptions for distributed implementation using IoT devices, and explore their drawbacks. We then discuss the implementation challenges and issues when applying FL to an IoT environment. We highlight an overview of FL and provide a comprehensive survey of the problem statements and emerging challenges, particularly during applying FL within heterogeneous IoT environments. Finally, we point out the future research directions for scientists and researchers who are interested in working at the intersection of FL and resource-constrained IoT environments.**

***Index Terms*—Convergence, federated learning (FL), global model, local model, on-device training, resource-constrained Internet-of-Things (IoT) devices.**

I. Introduction

IN THIS section, we explain the motivation to conduct a comprehensive survey on federated learning (FL) for resource-constrained Internet-of-Things (IoT) devices, followed by recently published prior works, and differentiate how our proposed survey is necessary for the FL domain. After that, we discuss our contributions and the necessity of conducting this research. Finally, at the end of this section, we briefly highlight the organization of this article.

A. Motivation

The ever-growing data collected/produced at edge devices are the result of billions of connected IoT devices as every active IoT client extracts their observed data and pushes those data to the edge. The traditional machine learning (ML) approaches need to perform aggregation of that extracted data element on a data center or a single machine, and such a learning scheme is common in different AI-based giant companies such as Facebook and Google. Companies store all the data collected in their data center, where they train the respective ML model. To attain a better ML model under the conventional centralized approach, the users may need to compromise their privacy by sending private data to the data center. Such a model training strategy is privacy intrusive, particularly when the clients need to address their personal or sensitive data to achieve a better training model.

FL is such an approach that is capable of training a model, leveraging the private data of clients without ever sharing it with other entities. However, the client may possess a lack of resources to perform on-device computation and may fail to reach the target convergence within an expected time. Moreover, we may face some unique challenges that could not be observed in a traditional FL-based approach in terms of communication, computation, privacy, storage, power, and energy utilization, e.g., straggler issue, high energy consumption, handling dropped participants. This article reveals the challenges of FL setting in such a situation and describes the impact of having such resource-constrained clients within a network by considering their practical constraints. To that end, we emphasize the open research issues in this area and enumerate numerous future directions.

B. Related Works and Contributions

Numerous studies from a wide range of research disciplines, including databases, distributed systems, cryptography, ML, and data mining, explored FL methods from various perspectives. It is a prevailing goal to learn from the distributed data set and simultaneously preserve privacy by not exposing the data. In 1982, a cryptographic mechanism was developed to apply on encrypted data [1]. The works of [2]–[6] are some of the early examples to discover knowledge from local data while maintaining privacy. To that end, the introduction of

Manuscript received February 28, 2021; revised April 29, 2021; accepted June 29, 2021. Date of publication July 6, 2021; date of current version December 23, 2021. *(Corresponding author: M. Hadi Amini.)*

Ahmed Imteaj and M. Hadi Amini are with the Knight Foundation School of Computing and Information Sciences and the Sustainability, Optimization, and Learning for Interdependent Networks Laboratory, Florida International University, Miami, FL 33199 USA (e-mail: moamini@fiu.edu).

Urmish Thakker is with Deep Learning Research, SambaNova Systems, Palo Alto, CA 94303 USA.

Shiqiang Wang is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA.

Jian Li is with the Department of Electrical and Computer Engineering, Binghamton University, State University of New York, Binghamton, NY 13902 USA.

Digital Object Identifier 10.1109/JIOT.2021.3095077

FL maintains privacy by storing client data only on-device, eliminates the dependency on a single server to generate a prediction model by performing computation on client devices, and builds a smarter model by learning from various client models. It is to be noted that any resource-constrained device could be a server in an IoT environment; therefore, it is not a good solution to consider such a device to store all the extracted data of the available clients and generate a model like conventional ML approach. Instead, the server can only be used to perform aggregation on the collected local models to generate an updated global model. In this article, we focus on the deployment and implementation of FL in an IoT environment, where the IoT nodes are considered as clients with limited resources. These resources include computation power, communication bandwidth, memory, and battery power. The IoT clients may have different technical characteristics and available resources, and that is why all the clients cannot be treated the same. The list of abbreviations that are used in this article is listed in Table I.

TABLE I
LIST OF ABBREVIATIONS USED IN THIS ARTICLE

Abbreviation	Description
CM	Cryptographic Method
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
DP	Differential Privacy
DRL	Deep Reinforcement Learning
DT	Decision Tree
FedAvg	Federated Averaging
FL	Federated Learning
FLS	Federated Learning System
GAN	Generative Adversarial Network
IID	Independent and Identically Distributed
IoT	Internet of Things
LM	Linear Model
LSTM	Long Short Term Memory
MEC	Mobile Edge Computing
ML	Machine Learning
MLP	Multilayer Perceptron
NLP	Natural Language Processing
NN	Neural Network
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TFF	TensorFlow Federated
UE	User Equipment

Several detailed surveys on FL have already been conducted by assuming that all clients within the network are resource unbounded. Li *et al.* [7] presented an overview of challenges, open problems, and issues associated with FL by considering the heterogeneity of devices; however, they assumed that all clients are resource-boundless. Yang *et al.* [8] focused on the categorization of FL settings while Niknam *et al.* [9] presented the issues of FL in a wireless environment. Besides, a federated optimization-based framework is proposed in [10], which is constructed by addressing challenges related to both system and statistical heterogeneity. They mentioned that straggler client is responsible for increasing statistical heterogeneity which put adverse impact during convergence. By adding proximal terms during local training, they obtained faster convergence and were able to analyze the effect of heterogeneity. Another exciting paper [11] discussed FL from the perspective of mobile-edge computing (MEC), including a caching and communication mechanism at the edge, while a detailed survey is presented in [12] by analyzing the recent advancement and issues of FL.

There are a couple of survey papers available on FL systems (FLSs), and we listed them in Table II. The papers in Table II are classified according to the area of FL and edge computing. FL survey papers [7], [7]–[12] are mostly focused on FL settings, system design, and components, implementation challenges, or on recent advancements. On the other hand, edge computing surveys [13]–[19] are mainly conducted on edge computing infrastructure, applications including ML and AI, resource management, wireless communication, and security and privacy issues. However, all these works considered only heterogeneity of systems or their statistical data, and did not discuss the challenges that would arise when the clients are resource bounded. Throughout this article, we point out the FL challenges while applying on a resource-constrained IoT environment, analyze the potential solutions toward those challenges, and reveal the future directions of this domain. This article is mainly a critical survey on the previous works that identifies gaps in resource-constrained FL implementation. To the best of our knowledge, this article is the first comprehensive survey on FL for resource-constrained IoT devices.

C. Organization

The remainder of this article is organized as follows. In Section II, we present an overview and taxonomy of FL with a comprehensive list of existing studies. In Section III, we review distributed optimization and ML approaches. Section IV presents a detailed analysis of the major challenges of FL while applying on resource-constrained devices, which is followed by Section V, where we discuss the potential solutions of those emerging challenges. After that, in Section VI, we present the existing FL applications, and in Section VII, we highlight the future research direction in the FL-based IoT domain. Finally, in Section VIII, we conclude our article.

II. OVERVIEW OF EXISTING STUDIES ON FEDERATED LEARNING MODELS

This section covers the definition of FL, a detailed description of the FL taxonomy, a brief highlight on the existing FL frameworks, and a comparison summary of the existing FL-based studies, which are classified in terms of privacy maintenance, attack schemes, fairness, learning effectiveness, and resource utilization.

A. Definition of Federated Learning

FL can be defined as a distributed ML approach, where the clients train themselves locally without sharing their direct information to the server. By periodically updating a shared

TABLE II
COMPARING OUR PROPOSED SURVEY PAPER WITH EXISTING SELECTED SURVEYS ON FL AND MEC

Reference	Area	Published in	Year	No. of Times cited (as of 2/25/21)	Contribution	Considered FL for resource-constrained IoT?
[7]	FL	IEEE Signal Processing Magazine	2020	390	Tutorial on characteristics and implementation challenges of FL	No
[8]	FL	ACM TIST	2019	729	Discussed categorization of FL settings e.g. vertical, horizontal	No
[9]	FL	Arxiv	2019	76	Presented applications and issues of FL in wireless communications	No
[10]	FL	IEEE Communications Surveys & Tutorials	2020	197	Comprehensive survey on FL in mobile-edge computing	No
[11]	FL	IEEE Network	2019	227	Discussed on MEC, caching, and communication of FL	No
[12]	FL	Foundations and Trends® in Machine Learning	2021	456	Survey on recent advancements and open problems of FL	No
[13]	Edge Computing	IEEE	2019	289	Survey on AI for edge intelligence	No
[14]	Edge Computing	IEEE Internet of Things Journal	2017	834	Survey on edge computing infrastructure, applications, security and privacy issues	No
[15]	Edge Computing	International Journal of Machine Learning and Cybernetics	2018	54	Survey on ML applications for IoT	No
[16]	Edge Computing	IEEE Communications Surveys & Tutorials	2017	1804	Survey on resource-management in mobile edge computing and wireless communication	No
[17]	Edge Computing	IEEE Network	2018	662	Survey on DL for IoTs in MEC environment	No
[18]	Edge Computing	IEEE Access	2017	593	Survey on computation, caching, and communication strategies at mobile edge	No
[19]	Edge Computing	IEEE Communications Surveys & Tutorials	2017	1452	Survey on Computation offloading, resource-management and mobility management in edge computing	No

global model based on performing aggregation of each client model information, this approach trains each device to capture the global view [20]. A high-level architecture of FL process is presented in Fig. 1. The FL process generally includes three steps:

Step 1 (Initiate Training Task and Global Model): In the initial phase, the central server decides the task requirement and target application. A global model (W_G^0) is initialized and the server broadcasts that global model to the selected local clients that are known as participants.

Step 2 (Local Model Update): Each participant generates a model utilizing their local data. Upon receiving the global model W_G^t (where t denotes the tth iteration), each client k updates its model parameters W_i^t for finding optimal parameters that minimize the local loss function $F_k(W_k^t)$. The local optimal models are then shared with the FL server.

Step 3 (Global Aggregation): After receiving the local models from the participants, the FL server performs aggregations and generates an updated global model (W_G^{t+1}). The latest global model is again shared with all the new participants.

Steps 2 and 3 are repeated until the central server reaches a convergence by minimizing the global loss function $F(W_G^t)$, which can be expressed as follows [21]:

$$\min_w f(w) = \sum_{k=1}^{N} P_k F_k(w)$$

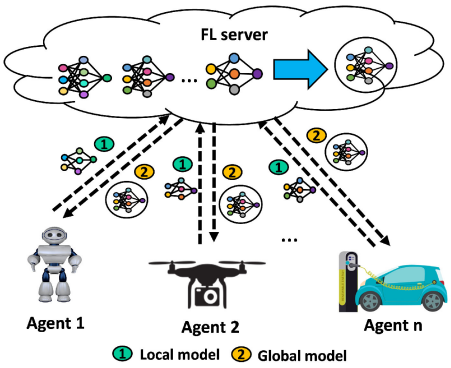


Fig. 1. FL procedure considering N number of participants.

where N is the total number of available devices, $P_k(\geq 0)$ indicates the relative impact of each device k while satisfying $\sum_k P_k = 1$, and $F_k(w)$ is the expected prediction loss on a sample input of the kth device on parameter w. Each device k possesses n_k samples (where $n = \sum_k n_k$). Thus, the relative impact of each local device can be expressed as $P_k = (n_k/n)$.

B. Taxonomy of FL-Based Systems

FLS can be categorized according to data sample, communication, prediction model, scale, privacy, and participation motivation (see Fig. 2). In this segment, we discuss each individual categorization instance with proper examples.

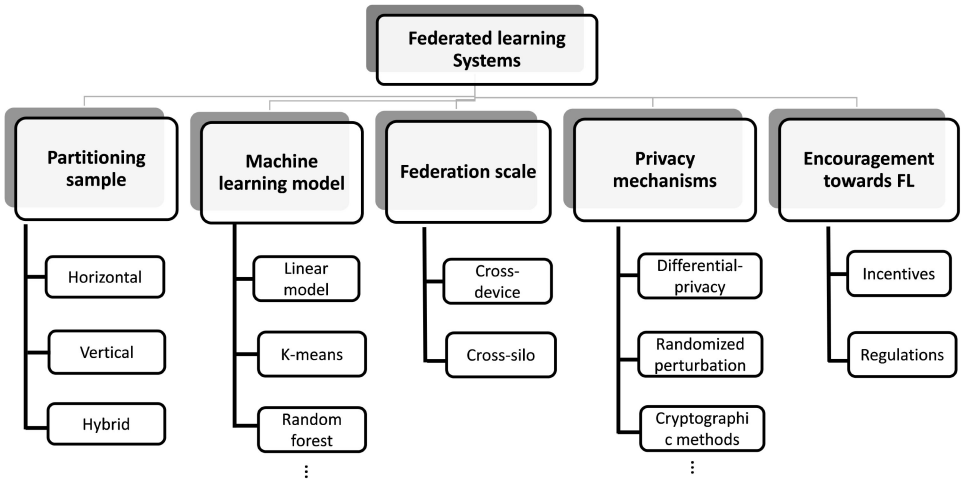


Fig. 2. Taxonomy of FL-based systems.

1) Partitioning Sample: While designing an FL model, we need to analyze the data distribution records by utilizing both the features and nonoverlapped instances. We can categorize FL into: 1) Horizontal FL; 2) Vertical FL; and 3) Hybrid FL based on the data samples distributed over networks and feature space of those samples.

a) Horizontal FL: Horizontal or sample-based FL have different data samples, but they share the same feature space. In Fig. 3, we can see that two client devices have a data sample that is generated using some similar applications, and each client device has an identical feature space. Each client generates a local model by utilizing the data samples and carry out the FL process. We can also consider horizontal FL from the perspective of real-life scenarios. Assume that two local superstores have different customers; thus, the user intersection set would be minimal. However, the business structure and policy of the two superstores may be similar, i.e., the feature spaces are aligned. In such a case, we can apply horizontal FL to perform the learning action. Most of the FL studies conform to the horizontal FL strategy, where the local participants train their model by sharing the same feature space, and a similar global model architecture is generated. Next-word prediction [22], wake-word detector [23], and recommendation system [24] are some examples of horizontal FL.

b) Vertical FL: In vertical or feature-based FL, the data sets share different sample spaces, but the sample IDs are the same (see Fig. 4). For instance, consider a bank and a superstore in the same area. Most of their customers may be the same, but their business structure, i.e., the feature space, is different, and thus the user-space intersection is quite large. We can consider another example. Assume that we want to make a prediction model for product purchases based on user information, credit card rating, and purchasing history. In such a case, vertical FL can perform aggregation of these different features and collaboratively construct a prediction model. SecureBoost [25] and FedBCD [26] are some of the examples of vertical FL.

c) Federated transfer learning: Federated transfer learning (FTL) [27] can be considered as the combination of

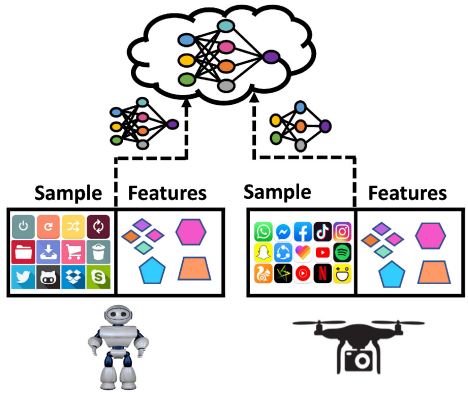


Fig. 3. Horizontal FL scenario.

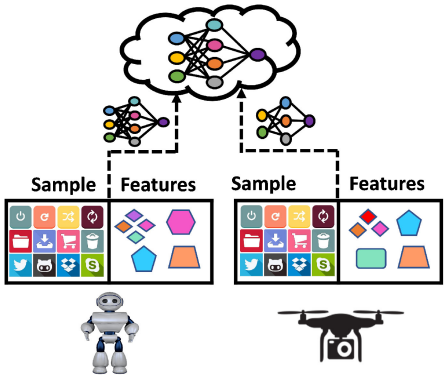


Fig. 4. Vertical FL scenario.

both horizontal and vertical partitioning of data (see Fig. 5). Horizontal and vertical FL would not be effective when two clients (*A* and *B*) have small overlapping data samples and feature space, and we need to learn all the sample labels of a client (e.g., client *A*). FTL is applicable in such scenarios, where the data samples and feature spaces are both different in the two clients' data sets. In other words, FTL can be applied when the clients' local data can differ in terms of both data samples and feature space. For instance, a group of research labs

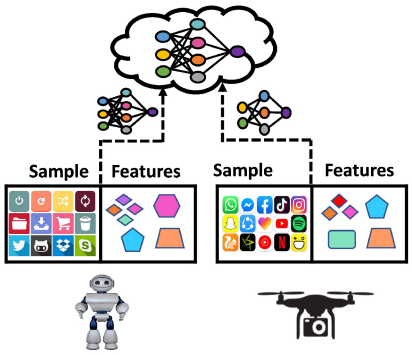


Fig. 5. FTL scenario.

wants to invent a COVID-19 vaccine, but their samples (e.g., testing samples may contain different coronavirus categories) and feature spaces (i.e., strategic plan and test results) may be dissimilar. Similarly, two different multinational companies located in different countries may have different customers (i.e., samples) as well as distinguishable rules and regulations (i.e., features). Due to geographical location difference of the two companies, the overlapping data sample would be negligible, while due to different business types, there may be a very small intersection in the feature space. In such a case, FTL can be applied to handle variance in data sample and feature space while performing on-device learning. In FTL, an overlapping representation between two feature space of the clients are learned utilizing the small common data samples and each client obtains predictions for local samples using one-side features. Liu *et al.* [28] designed a framework that can learn a feature representation of multiple parties based on common instances.

2) Machine Learning Model: The appropriate ML model needs to adapt based on the training objective. For instance, if we want to classify the objects from an image, we need to train the FL model using convolutional neural networks (CNNs). Several existing studies develop ML models for FL settings. The most popular ML model that is used in FL is federated stochastic gradient descent (Fed-SGD) coupled with the neural network (NN), e.g., image classification [20] and word prediction [22], [29]. The decision tree (DT) is another popular and widely used ML method that is highly efficient for training models. In tree-based FL, a model is generated for training single or multiple DTs. Cheng *et al.* [25] and Li *et al.* [30] designed gradient boosting DTs (GBDTs) by considering both horizontal and vertical partitioned data schemes. Different linear models (e.g., linear regression and classification, logistic regression, and support vector machine (SVM) [31], [32]) are convenient to handle. Such linear models are easier to learn than different complex models (e.g., DTs and NNs). In a nutshell, many FL applications and frameworks are proposed on FedSGD [20], [21], [33], [34]. SGD is basically a common optimization technique that can be applied in different models, including SVM, linear regression, and NN. To improve the model accuracy in a large-scale FLS and to cover the gap between FLS with state-of-the-art ML models, it is necessary to exploit the ML architecture for obtaining better FL training.

3) Federation Scale: FLS can be divided into cross-silo and cross-device categories based on the scale of federation [7], [12]. This categorization is performed based on the number of clients and their data quantity.

Cross-Device: In cross-device FL, the number of clients can be large, but each client has a limited size of data. Different smartphones or IoT devices can be considered as the clients of such a system, which could be millions or billions in number. Recently, Google has invented an FL-based keyboard suggestion [29] by training the model on-device of the user and aggregate the model information in the server. However, in such an approach, the clients may not be able to train themselves in a complex training environment because of resource scarcity. Thus, the server needs to be capable enough to process all the model information to generate a global training model.

Cross-Silo: Cross-silo FL holds a relatively small number of clients, but they own a large amount of data. Typically, in cross-silo FL, the clients are data centers or different organizations. For instance, Amazon recommends products by training models using the collected data from hundreds of data centers, where each data center stores large amounts of data and configured with sufficient computational resources.

4) Encouragement Toward FL: In real-world FL applications, the clients need encouragement or motivation to participate in the training phase and that can be carried out through regulations or incentives mechanism. For instance, Google FL keyboard suggestion [29] cannot force the users to provide data, but they ensure better keyboard suggestions to the users who upload their data. Such incentives motivate the users to share information or performing on-device training.

C. Summary of Existing FL-Based Studies

There have been several studies on FL due to its positive effect in terms of privacy preservation, resource utilization, and overall efficiency of the learning scheme [35]. We have extended the classification provided by Li *et al.* [35] and presented a detailed summary of those studies in Table III.

We provide the classification of the prior works based on two categories, i.e., FL algorithm (e.g., federated averaging (FedAvg) [20]), and feature integration (e.g., blockchain-based FL [36]). We categorized each work according to their data partition scheme, i.e., vertical and horizontal, as discussed in Section II-B1. For the sake of simplicity, different model types, i.e., NNs, linear models, and DTs, are abbreviated as "NN," "LM," and "DT," respectively. We provide the model types in some cases, where the authors applied multiple ML strategies for their proposed approaches. For instance, Truex *et al.* [37] combined differential privacy (DP) with multiparty computation to protect their system from inference threats and to generate high-quality models. We included all three model types (i.e., "NN," "LM," and "DT") for [37] as they validated their system using CNN, SVM, and DT. We also classified the existing studies based on their decision-making architecture. We realized that a major part of the existing approaches are based on server-centric design using trusted servers (e.g., [38]). However, such a server-centric decision-making architecture

may face trust issues, particularly in a cross-silo FL setting. In order to handle a cross-silo environment, one approach is to replace the central server while enabling each client to share their model parameters and maintaining a similar global model. Such strategies increase the computational cost and communication overhead as compared to the server-centric approaches. In Table III, we explore such studies (e.g., [39]) as "Client-centric" in the "Decision-making" column. We also considered the privacy methods that are deployed in prior works and categorized them in three groups: 1) cryptographic methods (CMs); 2) DP; and 3) hashing-based privacy. In CM, a cryptographic strategy is adapted to encrypt client data (e.g., the tree boosting model [25]) during communication or data storage considering the security threat involved in the learning process. DP is another privacy-preserving strategy, where the patterns of groups within the data set are shared while withholding raw data or information about any individual (e.g., [40]). Besides, some studies (e.g., [30]) considered a hashing-based privacy-preserving approach, where a hashing algorithm is applied on data to generate a hash that is used to verify data integrity. Finally, in the "Remark" column of Table III, we highlighted the main research directions (e.g., effectiveness, fairness, privacy, and incentives) of the existing FL studies.

III. Distributed Learning and Optimization Algorithms

In this section, we discuss the areas of research related to distributed learning and optimization techniques. Even though the main focus of the article is not in such domains, a brief highlight on these areas could motivate researchers to bring with a new or improved version of distributed learning setting or optimization techniques.

A. Federated Learning Algorithms

As we discussed earlier, after introducing FL by McMahan *et al.* [20], several modified versions of the algorithms are proposed that can be effective in different circumstances. In this segment, we present some of the well-known and effective FL algorithms that would motivate researchers to introduce an improved version of FL.

1) Federated Averaging: FedAvg algorithm [20] performs training operation via a central server that propagates a shared global model w_t, where indicates t the communication round. However, each client orchestrates local optimization using the concept of SGD. This algorithm has five hyperparameters: 1) a fraction of clients or participants C that takes part in the training round; 2) size of local minibatch B; 3) learning rate η; 4) number of local epoch on the client-side before updating of the global model E; and 5) a learning rate decay λ. The algorithm is presented in Algorithm 1. When the system starts, the global model parameter w_o is randomly initialized (line 1). At each communication round of the server, a fraction of clients is selected (line 3), and a random set of the client is chosen for the training phase (line 4). Each client sends his/her local optimal model parameter, which is then aggregated onto the server (lines 5–7). The iteration period continues until a certain number of iterations, or if the update is small enough, or reaches a convergence.

Algorithm 1: FedAvg [20]. The Index of N Clients Are Denoted by k; B Represents the Minibatch Size of Local Client, E Is the Number of Local Epochs, n_k Is the Local Examples of a Client While n Denotes the Total Data Points, P_k Stores a Client Data Samples, and η Represents the Learning Rate

1 Initialize w_o
2 **for** each round $t = 0, 1, 2, \ldots$ **do**
3 $\quad c \leftarrow \max(\lfloor C \cdot N \rfloor, 1)$
4 $\quad R_t =$ random set of c clients
5 $\quad$ **for** each client $k \in R_t$ in parallel **do**
6 $\quad\quad w_{t+1}^k =$ UpdateFromClient (k, w_t)
7 $\quad w_{t+1} \leftarrow \sum_{k=1}^{N} \frac{n_k}{n} w_{t+1}^k$
8 **UpdateFromClient** (k, w): // Run on client k
9 $\quad Batch \leftarrow$ (split $P_k$into batches of size B)
10 $\quad$ **for** each local epoch i from 1 to E **do**
11 $\quad\quad$ for batch $b \in Batch$ do
12 $\quad\quad\quad w \leftarrow w - \eta \nabla \ell(w; b)$
13 $\quad$ **return** w to server

Algorithm 2: Local GD [70]. $\eta > 0$ Represents the Learning Rate, t_p Denotes a Particular Communication Time, and g Indicates a Fixed Number of GD Steps

1 Initialize vector w_0
2 Initialize $w_0^k = w_0$ for all $k \in [N] \stackrel{\text{def}}{=} \{1, 2, \ldots, N\}$
3 **for** $t = 0, 1, 2, \ldots$ **do**
4 $\quad$ **for** $k = 1, 2, \ldots, N$ **do**
5 $\quad\quad w_{t+1}^k =$
$$\begin{cases} \frac{1}{N} \sum_{g=1}^{N} \left(w_t^g - \eta \nabla f_g\left(w_t^g\right)\right), & \text{if } t = t_p,\ p \in \{1, 2, \ldots\} \\ w_t^k - \eta \nabla f_k\left(w_t^k\right), & \text{otherwise} \end{cases}$$
6 $\quad$ **end for**
7 **end for**

2) Local Gradient Descent: Large-scale models are often constructed, and first-order techniques are applied to solve related problems as they scale well in terms of dimension and data size. One popular choice is to use the Local gradient descent (GD) approach, where the optimization process is divided into epochs. Each iteration initiates to perform averaging steps across available N devices. The rest of the other epoch does not involve any further communication. Each client device implements a fixed number of GD steps (declares from the average model) using their local function independently in parallel [70]. See the details in Algorithm 2.

3) FedProx: In FL settings, the clients may need to perform a nonuniform amount of tasks that can handle the negative effect of system heterogeneity. Still, too many clients' updates can diverge the overall methods in the results of underlying heterogeneous data. Li *et al.* [21] proposed an algorithm named *FedProx* that is particularly useful for the resource-constrained FL-based IoT environment. They enable variable

TABLE III
COMPARING THE EXISTING FL LITERATURE. *NN*: NEURAL NETWORKS, *DT*: DECISION TREE, *LM*: LINEAR MODEL, *DP*: DIFFERENTIAL PRIVACY, *CM*: CRYPTOGRAPHIC METHOD

Approach	Category	Data partition	Model type	Decision-making	Privacy	Remark
FedAvg [20]	FL-based Algorithms	Horizontal	NN	Server-centric	–	SGD-based
Residual FL [41]			LM			
Agnostic FL [38]			NN, LM			
FL SVRG [42]			LM			
FedProx [21]			–			
FedBCD [26]		Vertical	NN			NN-specialized
Bayesian FL [43]		Horizontal				
FedMA [34]						
Tree-based FL [39]			DT	Client-centric	DP	DT-specialized
SimFL [30]					Hashing	
FedXGB [44]				Server-centric	CM	
FedForest [45]						
SecureBoost [25]		Vertical				
Ridge Regression FL [32]		Horizontal	LM			LM-specialized
PPRR Linear Regression FL [46]						
Linear Regression FL [47]		Vertical				
Logistic Regression FL [31]		Horizontal				
Federated MTL [48]					–	Multi-task learning
Federated Meta-Learning [24]			NN			Meta-learning
Personalized FedAvg [49]						
Lifelong FL [50]						Reinforcement learning
FedNAS [51]						Efficiency improvement
Structure and sketched updates [33]	Functionality Upgrade					Efficiency improvement
Multi-Objective optimization FL [52]						
Federated distillation [53]						
FL STC [54]						
Client-Side DP FL [40]					DP	Privacy guarantees
FedSel [55]						
FL Language Models [56]						
Federated Extra-Trees [57]			LM			
FL Scalar DP[58]			LM, NN			
Secure Aggregation FL [59]			NN		CM	
SPC+DP FL [37]			LM, DT, NN		CM, DP	
Backdoor FL [60]			NN		–	Attacks
Adversarial Lens FL [61]						
Distributed Backdoor [62]						
Fair-allocation FL [63]			LM, NN			Fairness
FedCoin [64]			LM			Incentives
Blockchained FL [36]						
Contract Theory FL [65]						
Client Selection FL [66]			NN			Edge computing
Adaptive FL [67]			LM, NN			
LEAF [68]	Benchmark		–	–		benchmark
Revocable FRF [69]						

local updates from the participated devices by adding a proximal term within the local subproblems. The proximal term is useful in two aspects. First, it limits the client's local updates to address the statistical heterogeneity issue. Second, it helps incorporate a variant amount of clients to work safely. We summarize the technique in Algorithm 3.

Algorithm 3: FedProx [21]

for $t = 0, \ldots$ **do**
  Server randomly chooses a subset R_t of N devices (each client k is chosen with probability P_k)
  Server sends latest global model w_t to all chosen clients
  Each device $k \in R_t$ finds a w_{t+1}^k where, $w_{t+1}^k \approx \arg\min_w h_k(w; w_t) = F_k(w) + \frac{\mu}{2}\left\|w - w^t\right\|^2$
  Each device $k \in R_t$ sends w_{t+1}^k back to the server
  Server aggregation, $w_{t+1} = \frac{1}{N}\sum_{k \in R_t} w_{t+1}^k$
end for

4) q-FedAvg: Though the state-of-the-art FedAvg significantly accelerates the convergence speed [20], it fails to allocate client resources fairly (performs uniform allocation of resources). The allocation of resources is significantly crucial when we consider resource-constrained devices for the FL process. With this motivation, Li *et al.* [63] proposed the q-FedAvg algorithm that can impose fairness based on the clients' contributions. In the q-FedAvg algorithm, for given cost functions F_k and parameter $q > 0$ (which is the fairness amount we wish to impose), the FL objective is defined as

$$\min_w f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w)$$

where $F_k^{q+1}(\cdot)$ denotes $(q+1)$ as a power of $F_k(\cdot)$. Here, q is a parameter that tunes the amount of fairness we wish to impose. When $q > 0$, it prevents the execution of local SGD. To solve the issues of the local updating approach, particularly

Algorithm 4: q-FedAvg [63]

for $t = 0, 1, 2, \cdots$ **do**
  Server randomly selects a subset R_t of N devices (each client k is chosen with probability P_k)
  Server sends latest global model w_t to all chosen clients
  Each chosen client device k updates w_t by performing SGD for E epochs with η to obtain $\bar{w}_{t+1}^k$
  Each selected client k computes:
    $\Delta w_t^k = L\left(w_t - \bar{w}_{t+1}^k\right)$
    $\Delta_t^k = F_k^q(w_t)\Delta w_t^k$
    $h_t^k = qF_{q-1}^k(w_t)\left\|\Delta w_t^k\right\|^2 + LF_q^k(w_t)$
  Each selected client k sends his/her parameters Δ_t^k and h_t^k to the server
  Server update: $w_{t+1} = w_t - \frac{\sum_{k\in S_t}\Delta_t^k}{\sum_{k\in S_t}h_t^k}$
end for

while allocating resources, Li *et al.* [63] proposed a heuristic solution by replacing the gradient with the client's local updates obtained by running SGD on each local device. The algorithm is depicted in Algorithm 4.

B. Distributed Learning

As discussed in Section II, the central server of the FL process orchestrates the learning process by managing the contributions of its clients. Thus, it can be considered as a single point of failure [see Fig. 6(b)]. Though large organizations and companies may afford to place a powerful, robust, and secure central server to carry out the training process, all types of sectors cannot adapt that [71]. Besides, some clients within the network can slow down the overall process [72], [73]. The main idea of fully distributed and decentralized learning is peer-to-peer communications of the clients that eliminates the central server [see Fig. 6(c)]. In contrast, on-device training without learning from a server or its peers is shown in [see Fig. 6(a)]. In these figures, the communication topology looks like a connected graph, where each node represents a client, and the line between two nodes specifies a communication channel. In a distributed learning mechanism, each round corresponds to a local update by the clients and information exchange with peers. Though we do not have any global model or state as in the standard FL, still, we can design the process such that all clients reach a global solution through local models. The local models can be converged through on-device training and learning from their peers [12]. Fully decentralized SGD and other optimization algorithms are recently getting popular for scalability in large-scale systems [74] and decentralization of networks devices [71], [75]–[82]. Note that even in the decentralized distributed setting, a central authority may need that will be in charge of setting up system configuration, learning tasks, hyperparameters, algorithm selection, or resolve system failure. A degree of trust needs to establish among the clients to replace the central authority. Alternatively, such decisions can be made by a leader client, through a collaborative consensus scheme [56], [59], [83].

C. Distributed and Federated Optimization

The early trend of distributed optimization was naive distributed variants of corresponding serial algorithms, which is often inefficient in terms of communication. The second trend is to design communication-efficient algorithms. The idea is to perform a lot of local computation that is further followed by a communication round. Such a technique is useful in practice and distributed approximate Newton (DANE) [84], CoCoA [85], and DiSCO [86] are some of the examples of such distributed optimization techniques. In distributed optimization, the data-centers possess huge data with relatively few devices. Later on, federated optimization is introduced to protect privacy in a better way. In that concept, the users keep their data private and provide the computational power of resources. Consequently, the data points are relatively smaller, the number of devices is huge, and data patterns vary on different devices.

There have been various methods to deal with distributed online optimization and distributed learning, including stochastic variance reduced gradient (SVRG) [87], [88], DANE [84] that is particularly for distributed optimization, naive Federated SVRG, and Federated SVRG (FSVRG). The desirable properties while designing an algorithm for unbalanced, non-independent and identically distributed (IID), and massively distributed can be stated as follows [33].

1) An algorithm stays there in case it is initialized to the optimal solution.
2) In case a single node possesses all data, the algorithm should converge in $\mathcal{O}(1)$ communication rounds.
3) If all available features within the system occur on a single node, then the problem can be decomposed, and the algorithm is supposed to converge in $\mathcal{O}(1)$ rounds of communication.
4) If we assume that each node has an identical data set, then the algorithm converges in $\mathcal{O}(1)$ communication rounds.

Property 1) is valuable for any optimization setting whereas properties 2) and 3) are applicable in federated optimization systems (e.g., unbalanced, non-IID, massively distributed). To the end, property 4) is an extreme case, particularly for a distributed optimization setting where we have a large number of IID data per device.

IV. Learning on Resource-Constrained Devices

Before discussing the challenges associated with resource-constrained devices, it is essential to understand the definition of on-device learning of edge devices clearly. We can define an edge device as a resource-constrained entity with limited computational power, storage capacity, transmission range, and battery [89]. We consider an object as an edge device if it cannot be integrated with additional resources, i.e., the device resources cannot be increased or decreased. For instance, a workstation cannot be considered as an edge device as we can

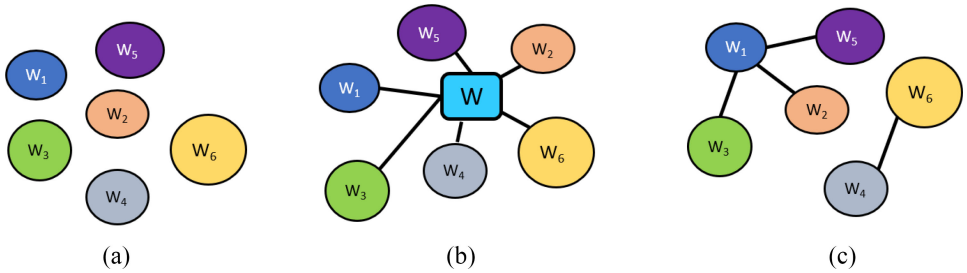


Fig. 6. Different modeling approaches in federated networks. Depending on properties of the data, network, and application of interest, one may choose to (a) learn separate models for each device, (b) fit a single global model to all devices, or (c) learn related but distinct models in the network.

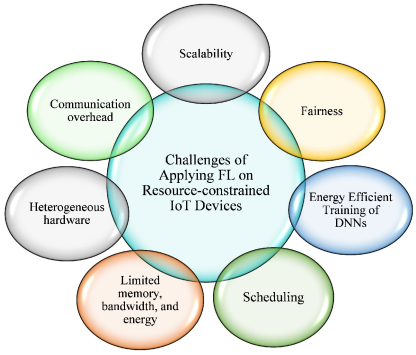


Fig. 7. Core challenges of FL considering resource-constrained IoT devices.

integrate additional resources within that device. However, a manufactured robot can be considered as an edge device since we cannot directly incorporate any more support to the robot's capability. If we look at our today's IoT world, then we can see the use of resource-constrained devices in every aspect, from monitoring the environment to controlling human life. Within such an IoT environment, edge devices are utilized as they are smaller in size and are more transportable. Different kinds of robots, drones, and smartphones can be considered as edge devices possessing limited resources that communicate remotely. To attain optimal service performance from such IoT components, we need to train those edge devices that prevent them from being stragglers during the learning process. Those devices should be trained with diversified sample environments to perform accurate prediction in a variety of testing data. It is not feasible to train those edge devices with a large data set due to their limited resource availability. In this section, we discuss the potential challenges we may face while considering such resource-bounded IoT nodes in the FL environment (see Fig. 7).

A. Communication Overhead

Communication overhead is considered one of the major challenges in an FL-based IoT environment. The communication cost mainly increases due to the large sizes of data passing during the process and the iterative and nonoptimized approach of conducting communication between the server and the clients. This problem becomes adverse when clients possess insufficient resources. For instance, if a client possesses limited bandwidth, then the client would not be able to communicate with the FL server effectively during model training. Similarly, if a client has weak processing capability, then performing an assigned local computational task would be infeasible for that client. Furthermore, there could be large data across the network that could produce a large model size, and eventually, the resource-constrained clients would struggle in dealing with such a large model. To carry out efficient training in a large data network, the client models need to be compressed so that the clients do not have to waste extra resources in training a large model. If a majority of the FL clients are resource constrained, then the FL process requires more server–client interaction to reach a target convergence, and the clients would not be able to afford such a high communication cost. While frequent FL server–client interaction can reduce convergence time, recurrent communication can encounter high costs. Therefore, it is required to design an effective optimization technique that can handle the tradeoff between communication overhead and resource utilization of the FL setting. Ma *et al.* [90] analyzed the tradeoffs of communication and resource expense; however, they did not address the complexity of the clients' local model solutions.

B. Heterogeneous Hardware

The training phase of FL can run on multiple devices, which may belong to various generations of products. Such product variation creates a network that consists of heterogeneous devices with a discrepancy in computational ability, memory size, or battery life. Therefore, the training period may vary significantly across clients, and it is not effective to consider all participants with the same scale. To achieve optimal results in training, FL needs to be aware of heterogeneous hardware configurations [7]. The proficient and trusted clients need to be selected in the training phase considering system requirements. After selecting suitable clients, it may be possible that a model fails to send its local model due to connection error or out of a battery issue (see Fig. 8). However, due to system requirements (e.g., memory and bandwidth), most of the clients may not be able to be a part of the training round. Besides, it is possible that the majority of proficient clients go out of networks, and we may end up with a few clients that do not satisfy system requirements. Thus, carrying out the FL training process in such a situation is challenging.

C. Limited Memory and Energy Budget

In Section IV-B, we discuss heterogeneous hardware challenges, and in this segment, we describe the memory

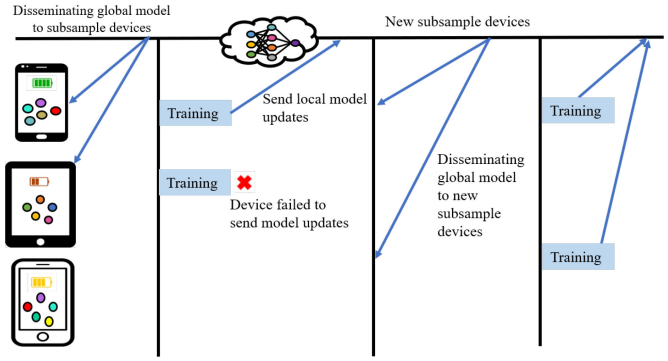


Fig. 8. Systems heterogeneity scenario in FL.

availability and energy budget issue across heterogeneous FL clients. Any FL client may have a very limited memory size, or a client having a larger memory size may not have space. Besides, the FL clients may have a preset energy budget, which may not fulfill the system requirements during the training process. While limited computational ability takes more processing time, memory shortage leads to overflooding of the device. Such situations encounter extra communication overhead and degrade the system performance. Hard *et al.* [22] pointed out the necessary hardware requirement, including the required memory size and processing ability during their implementation of next-word prediction on the keyboard. They mentioned that to simulate their application, the device should have a minimum of 2 GB of free memory, whereas many IoT devices hardly possess even free megabytes of memory. Considering such memory constraints, Haddadpour *et al.* [91] proposed an approach of distributing shards of data across FL clients to attain the target model swiftly. In their approach, they selected proficient clients who possessed a greater memory size, energy budget, higher bandwidth, and processing capability. However, they did not discuss the memory management and data handling for FL clients with limited available memory. We can manage such memory limitation by storing limited sizes of data, and in case of memory shortage, data aggregation technique can be applied to avoid memory outburst. Wang *et al.* [67], Das and Brunschwiler [92], Jiang *et al.* [93], and Xu *et al.* [94] analyzed hardware limitation challenges in the implementation of FL by considering Raspberry Pi and other types of resource-limited clients. They studied the feasibility of implementing FL on resource-constrained edge devices but did not cover the way of leveraging optimal memory requirement and quantifying energy budget throughout the FL process.

D. Scheduling

Existing federation optimization techniques can be classified into synchronous and asynchronous training. McMahan *et al.* [20], Konečnỳ *et al.* [33], [42], and Bonawitz *et al.* [73] focused on analyzing federated optimization that considers synchronous communication during training between the FL server and clients. In every training round, a subset of clients is triggered to perform a task. However, device or network issues can compel some clients to be unresponsive in the process, and the server needs to wait until getting a response from sufficient clients. Otherwise, the server drops that epoch as time-out and proceeds on to the next iteration. On the contrary, asynchronous optimization enables FL participants to directly send gradients to the FL server after every local update that is excluded in synchronous FL optimization. Asynchronous training [95] is applied in some recent works because of its faster convergence when communication latency is comparatively higher and heterogeneous across the clients. Xie *et al.* [96], Zheng *et al.* [97], and Lian *et al.* [98] analyzed asynchronous FL training with provable convergence by combining it with federated optimization. We present the synchronous and asynchronous FL behaviors in Fig. 9.

In an FL process, it is indispensable to set the training phase of the participants, which is called scheduling. Scheduling is explicitly important when there exists resource-constrained IoT devices within the networks, and frequent interaction with the server costs more resources. Optimized scheduling can play a vital role in minimizing energy consumption as well as utilizing less bandwidth. Scheduling should be carried out in such a manner so that there remains less possibility of possessing old data by the participants. It is possible that some participants can generate local models utilizing their old data repeatedly while skipping new data [73]. Such a situation can lead to resource-wastage without bringing any variations or improvements in the model. Besides, any participant can collect data by using a malicious application, and recognizing such harmful application data can be a challenge as it needs extra resources. Moreover, improper scheduling can lead to slow learning or straggler issues, which is considered as one of the reasons of a performance bottleneck, particularly for a resource-constrained FL-based IoT environment. By straggler clients, we mean the IoT devices that fail to respond within a specified period while the other clients react to the server successfully. Due to the slow response, the server needs to wait for the straggler client model, resulting in a delay in performing the aggregation of the model parameters in synchronizing FL. If the number of such straggler clients is high, then the overall model convergence would be at stake [99], [100]. Besides, in the conventional FL approach, the straggler clients are simply dropped [101]. However, if a significant portion of the clients is straggler, and we drop all of them, then the model quality would be extremely low [94]. Therefore, it is challenging to leverage a proper scheduling and guarantee model convergence even when a major portion of the clients are stragglers.

E. Energy-Efficient Training of DNNs

Deep NNs (DNNs) are applied in various artificial intelligence and deep learning-based applications where the sample data set is large. A lot of edge applications are now using DNN-based algorithms [102], [103] and there is an increasing focus on making DNN inference efficient on edge devices [104], [105]. Additionally, FL requires edge devices to perform on-device training. However, training DNNs requires high computational capability, large memory, and energy availability, and most of the IoT clients may lack such system

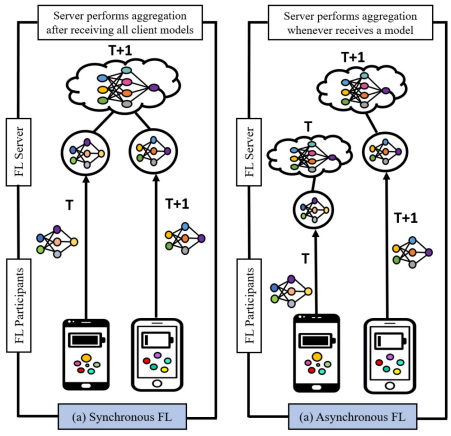


Fig. 9. Difference between a synchronous and asynchronous FL.

configurations. Wu *et al.* [106] proposed an approach to reduce the cost of training and inference by using lower bitwidth integers for both stages of the application. Jiang *et al.* proposed an efficient learning technique using pruned models, while Park *et al.* [107] proposed a strategy to generate a high-quality ML model through on-device model output, parameters, and data aggregation. Another interesting approach to training high capacity models having fewer parameters is discussed in [23]. Specifically, in case the size and features of the training data set are huge, we need to devise energy-efficient training on resource-constrained clients, perhaps a challenging approach.

The memory requirement of the training phase of DNN has been an issue that is well studied for training on large GPU and CPU server clusters. Training on edge devices can benefit by adapting techniques that have served well in managing this problem in the server context, such as efficient gradient checkpointing [108], tensor rematerialization [109], and recompute [110]. Cai *et al.* [111] proposed another way to reduce the memory required for on-device training by introducing a lite residual module that can be adapted to new data. By only changing this lightweight module and keeping the other parameters constant, they reduce the memory requirement of the training process.

Another factor that can cost extra energy resources during training is mislabeled or unlabeled training data, particularly when the size of the data set is large. The existing FL-based applications consider that all extracted data are appropriately labeled. Nevertheless, this assumption can be disproved if the collected data are mislabeled via security holes or unlabeled due to a network connection error. Mislabeled data would generate a wayward model that eventually affects the global model update. In case we have unlabeled data, it costs extra resources to put labels, and that would be crucial for resource-constrained IoT settings. Gu *et al.* [112] proposed a framework to identify the mislabeled data which are injected through data poisoning attacks. Using representation-based fingerprints, they detect the malicious or compromised participant's data label while coming across erroneous predictions during runtime. Tuor *et al.* [113] proposed a method of finding and ignoring irrelevant data (possibly due to mislabeling) from FL. To come up with a solution of unlabeled data, Lim *et al.* [114] proposed a strategy to make labeling of unlabeled data through applying collaborative learning with the neighbor clients. Implementing the same procedure for resource-constrained clients would be challenging in real time as it needs additional resources.

F. Fairness in Federated Learning

Fairness in the FL process means the distribution of client resources in an equitable manner. We can think of the global model as a resource, which is responsible for serving the client devices. However, the service that each user receives needs to be fair, i.e., the resource allocation and accuracy distribution across the client devices are unprejudiced. A minimax optimization framework named Agnostic FL [38] is developed, which can optimize the target distribution of the centralized model and is formed as a mixture of participated client distributions. However, their proposed approach is applied only at small scales. Li *et al.* [63] used a α-fairness metric and proposed a q-Fair FL to ensure fair accuracy distribution. Their proposed strategy can tune resource distribution by considering the desired amount of fairness. A collaborative fair FL framework is proposed in [115], which utilizes client reputation and compels them to converge to different models. They achieved fairness without degrading predictive performance. In [116], an FL-based client selection process is investigated to minimize clients' model exchange time that guarantees long-term, flexible fairness in the presence of rigid system constraints. However, they could not figure out a way to quantify how the fairness factor would impact the convergence speed and final target accuracy.

Moreover, some recent works on the optimization of resource allocation with incentive mechanisms for the FL process can be found in [116]–[121]. Khan *et al.* [117] designed an incentive-based FL model via a Stackelberg game for motivating client participation in the learning process. With the motivation of addressing issues related to costs and mismatch between client's contributions and receiving incentives, Yu *et al.* [116], [122] proposed a payoff-sharing scheme named FL Incentivizer (FLI). Their proposed scheme can dynamically distribute a given budget among data owners by ensuring maximization of collective utility and minimization of inequality, considering the received rewards and waiting time for receiving those rewards. A trust and incentive-based FL model is designed in [119], where they proposed to add local computation results of the clients using the concept of blockchain consensus to establish a public auditable and decentralized FL ecosystem. In their model, honest clients can receive incentives while malicious clients are punished heavily in terms of payoffs. Besides, Nishio *et al.* [120] proposed a strategy of estimating the contributions of each client in an FL process and provide incentives accordingly, reducing the communication and computational overhead. Similarly, Cong *et al.* [121] designed a client contribution-based incentive method for FL but using the concept of Vickrey–Clarke–Groves (VCGs) mechanism.

After analyzing the above-mentioned FL-fairness strategy, we can conclude that any client within an FL-IoT

environment may have resource scarcity. Therefore, designing a fair resource allocation and distribution scheme is necessary to reduce communication overhead, computation power, and to achieve higher accuracy. We need to check clients' activities, resource status, and contributions toward model convergence to ensure fairness in FL resource allocation.

G. Scalability of Federated Learning

In a realistic FL-based IoT environment, we may observe a large number of IoT devices that are heterogeneous in nature and possess limited resources. In such a situation, FL training can be executed through effective client selection and optimal resource utilization. Chen *et al.* [123] developed a framework via joint learning and establishing wireless communication among the FL clients. They discussed that the FL process can be hampered due to packet errors or the unavailability of wireless resources (e.g., limited wireless bandwidth). Considering the factors, they formulate an optimization problem considering joint learning, resource block allocation, and effective user selection with a goal of minimizing FL loss function. They derive a closed-form expression for FL convergence by considering the effect of the wireless channel. Their proposed framework ensures scalability and sparsification. Nishio and Yonetani [66] design a client selection protocol, using FL edge server. Their proposed model can manage the communication resources between the FL server and the clients, choosing clients based on their resource conditions. Besides, an activity and resource-aware FL model is presented in [124]. They proposed a strategy of examining client's resources and assigning trust scores to clients as per their contributions toward model convergence. On the basis of sufficient resources and a higher trust score, they only select a subset of eligible clients for the training round from a large number of available clients. Their proposed model ensures scalability, robustness, and sparsification of the FL process. Ye *et al.* [125] proposed a selective client model aggregation-based FL framework for vehicular edge computing. Instead of a random selection of FL clients, they leverage a technique of selecting clients based on contract theory. Moreover, a trilayer lightweight FL framework is proposed in [126] that is capable to handle a large number of clients and their huge data streams across the networks. They shrink large model size through pruning mechanism, select clients based on their resource status and previous activities, handle divergent local model update, and also allow a variable local model update. Their proposed framework ensures scalability, quantization, robustness, and sparsification.

H. Privacy Issues

In federated settings, we keep the raw data of each client on-device due to privacy concerns. However, it is possible to leak sensitive information [58], [127]–[129] through sharing model update during the training process. For instance, Carlini *et al.* [128] presented that sensitive patterns (e.g., credit card numbers) can be extracted from a user-trained model based on recurrent NNs. In the case of having sensitive data sets distributed across several data owners, privacy can be preserved via secure multiparty computation (SMC) or secure function evaluation (SFE). The protocol outcome enables multiple data owners to collaboratively agree to generate a function without leaking any information [130]–[132]. Though several privacy definitions for FL are stated in [40], [56], [58], [128], and [133]–[138], we can classify them as global and local privacy. In the global privacy setting, the server is assumed to be trusted, and local model updates are private. In local privacy, individual local model updates are generated on the client side and aggregated on the server. In Table IV, we show key ideas of some existing FL-based privacy-preserving approaches. However, due to the presence of resource-constrained devices, the existing privacy-preserving FL algorithms may not be suitable for running on those devices. Thus, beyond ensuring rigorous privacy guarantees, novel methods need to be designed that are communication efficient, computationally cheap, and capable of handling dropped participants.

V. Potential Solutions of Emerging Challenges in Deploying Federated Learning Algorithms on Resource-Constrained IoT Devices

In the previous section, we explored the implementation challenges of the FL process during on-device training with resource-scarce devices. A clear direction toward possible solutions for those emerging challenges can be effective in future research of this domain. This section describes the existing works and possible solutions of emerging challenges during training of resource-constrained devices in an FL environment.

A. Deploying Existing Algorithms to Reduce Communication Overhead

We explored a couple of key approaches that aim to reduce communication costs and can be classified into three categories: 1) decentralized training; 2) model compression; and 3) importance-based updating. The integration of such strategies can be useful to overcome the tradeoffs and shortcomings in this area. Haddadpour *et al.* [91] proposed an approach to infuse redundancy among the clients to bring diversity and reach convergence taking less communication round. Chen *et al.* [123] also designed a framework of joint-learning by considering the effect of wireless factors on participants in the FL scenario. Some of these methods adapted model compression strategies, but those methods may deteriorate model accuracy and encounter high computational costs. Such tradeoffs are empirical, i.e., we need to conduct several local training rounds to find an optimal number of iterations before making a communication. FL method can be more scalable if we can apply effective optimization techniques that are formalized theoretically, and implemented and tested empirically. Apart from compressing the model size, FL approaches can be motivated by MEC paradigms and their applications. For instance, Liu *et al.* [146] considered an intermediate model aggregator for reducing instances during device-cloud communication. However, their model costs more time to converge when the number of clients or edge servers increased. The situation becomes adverse when there exist non-IID data across

TABLE IV
EXISTING FL-BASED PRIVACY PRESERVING APPROACHES

Approaches	Ref.	Key ideas
Client-side DP	[40]	Adapting dynamic DP-preserving technique during decentralized training.
mGAN-AI	[140]	Designed a framework by incorporating GAN with multi-task discriminator to differentiate category, and client identity within input samples.
No-Peek	[141]	A survey on differentially private FL-based privacy-preserving techniques.
Hybrid FL	[37]	Combine SMC with DP to obtain stronger privacy and produce a model with high accuracy.
NbAFL	[142]	Designed a framework based on DP to add artificial noises at the client side before performing model aggregation, and analyze algorithms and performance of DP in FL.
PEFL	[143]	Presented a privacy-enhanced FL framework to protect local model update in presence of an untrusted server using Paillier homomorphic cryptosystem.
Wireless DP	[144]	Analyzed how superposition properties of wireless can be beneficial for privacy.
Fog FL	[145]	Enabled IoT data to satisfy ε-DP to prevent data and model attacks.
TEE FL	[146]	Based on Trusted Execution Environment (TEE), designed a training-integrity protocol that can preserve privacy and ensures integrity in deep learning processes.

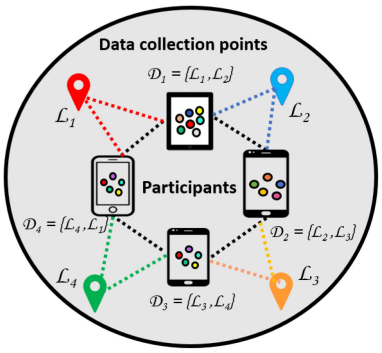


Fig. 10. Infusing data redundancy through overlapping data collection.

the network. Through multitask learning [48], such a statistical challenge can be handled. Moreover, FL models can be exploited to efficiently utilize the storage and computing power for facilitating efficient FL.

To reduce communication overhead, Imteaj and Amini [147] and Sun *et al.* [148] discussed infusing redundancy among the client data set to reach convergence with fewer communication rounds. In Fig. 10, we see that a particular data collection point $\mathcal{L}_1$ is used by two clients $\mathcal{D}_1$ and $\mathcal{D}_4$. Similarly, other data collection points, i.e., $\mathcal{L}_2$, $\mathcal{L}_3$ and $\mathcal{L}_4$ are utilized by $\mathcal{D}_1$ and $\mathcal{D}_2$, $\mathcal{D}_2$ and $\mathcal{D}_3$, and $\mathcal{D}_3$ and $\mathcal{D}_4$, respectively. This setting leads to infusing redundant data samples among the client devices.

According to [33] and [149], a novel approach is proposed to share compressed sizes of message and carry out the reduced number of communication rounds to attain the target model. With the same motivation, Caldas *et al.* [150] applied lossy compression and federated dropout to train a smaller subset of local clients and reduce client-to-server interaction and local computation (see Fig. 11). Though frequent communication may accelerate convergence, recurrent interaction incurs more communication costs. Every time a client interacts with the server, it has to compromise its resources. To handle the limited resources of the clients, a resource-optimization algorithm is necessary to consider such a tradeoff. In this regard, the authors in [26], [90], [151]–[155] studied the relation between communication cost and effective resource utilization, though they did not discuss the complications of the local problem's solution. Wang *et al.* [67] presented a distributed control algorithm for minimizing the training loss under a given resource budget. Besides, Wang *et al.* [11] designed a framework to exchange learning parameters of the clients through the collaboration for generating better local training models. Hence, this reduces communication overhead and ensures both system and application level improvement, which generates additional energy cost. A detailed discussion on the tradeoffs between FL training period and energy requirement cost can be seen in [114] and [156]–[158]. They minimized the weighted sum of the training completion period and energy consumption, applying an iterative algorithm. In the case of delay-sensitive scenarios, they adjusted the weights so that FL participants would expend more energy to achieve time minimization.

However, most of the studies that we discussed do not consider the heterogeneity of client resources. Due to such heterogeneity, some of the approaches cannot be adapted in such a resource-constrained FL-based IoT environment. For instance, the key idea of [20] was to allow for more computation on the mobile-edge side, e.g., by conducting more local updates before interacting with the server. Such an application requires processing power, which may not be feasible for IoT clients with weak-processing units. Finally, the resource-limitations may cause a straggler effect.

B. Convergence Guarantee in Asynchronous FL

In Section IV-D, we highlighted the difference between synchronous and asynchronous communication. Most of the existing FL approaches are implemented on the concept of synchronous FL, where the global model aggregation depends on the receiving of all the local model parameters of the participants. Previous works obtained fast convergence in such synchronous FL procedures, as they assumed all participants have sufficient resources (e.g., computation, bandwidth, memory). In consequence, even the slowest participant does not affect much the overall accuracy; and eventually, the model convergences. On the other hand, in asynchronous FL, the server performs aggregation whenever a model is received and may include a participant in the middle of the training phase. This approach enables scalability within the system and reduces the straggler impact, but cannot guarantee convergence. In Table V, we point out

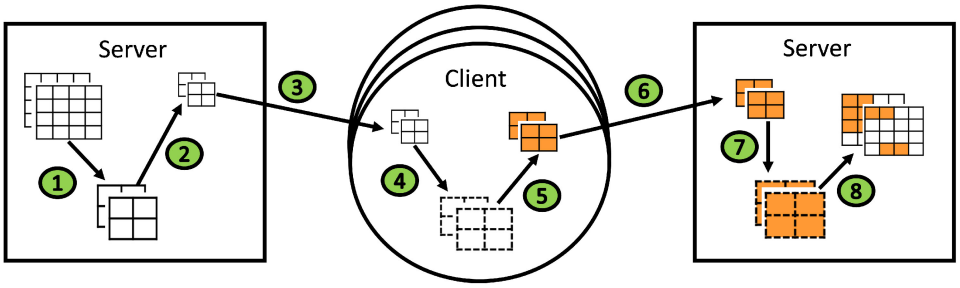


Fig. 11. Reducing size of the model by (1) generating a submodel applying Federated Dropout, (2) lossily compressing the obtained resulting object which is passed to the client, who (4) applies decompression and trains that using its own local data, and (5) again compresses the update which is sent back to network server. There it is (7) again decompressed and finally, (8) aggregated to be a part of the global model.

the key ideas of different asynchronous FL-based approaches. Sprague *et al.* [159] analyzed the issues of ensuring convergence of asynchronous FL but did not present a solution to overcome such issues. Xie *et al.* [96] proposed asynchronous federated optimization, [159] discussed on the asynchronous FL for geospatial applications, [160]–[162] proposed a DP-based asynchronous FL strategy for MEC, and [163] presented a blockchain-based secure data sharing strategy for asynchronous FL. Still, none of these works guarantee convergence during asynchronous FL communication of resource-constrained clients. Thus, formulating a method to ensure convergence in asynchronous FL can be a new research direction.

C. Quantification of Statistical Heterogeneity

FL training becomes complicated when the training data across devices are not identical in terms of data modeling and convergence behavior of the training process. Several ML works focused on the designing of statistical heterogeneity via meta-learning [166], [167], multitask learning [168], [169], which are further extended to FL settings [24], [48], [170]–[173]. For instance, an optimization framework MOCHA [48] allows for personalization through multitask learning; however it considers convex objectives and is limited to its ability while scaling to massive networks. Corinzia and Buhmann [171] modeled a Bayesian network by performing variational inference during learning. Though their proposed approach can handle both convex and nonconvex models, it encounters high cost while generalizing to large federated networks. Besides, Eichner *et al.* [172] aimed to identify cyclic patterns within data samples, while a detailed analysis of transfer learning strategy for personalization during FL can be seen in [170]. While the client data tend to be heterogeneous in terms of the number of samples, data set structure, and format in a non-IID setting, all existing works on FL adjust the statistical heterogeneity after the training phase begins. It impacts the training quality, and the lack of proper quantification of such heterogeneity can cause poor training performance. A local dissimilarity approach is proposed [174] to quantify statistical sample variation, where the resource-quantification starts after the training starts. Li and Wang [175] proposed a centralized approach of handling heterogeneity of FL model training but did not consider specific support and analysis for statistical heterogeneity. Li *et al.* [21] proposed a reparametrization of the FedAvg [20] algorithm that can scale up divergent model updates and guarantees convergence while learning over statistical heterogeneous networks. However, they did not quantify the level of statistical heterogeneity while selecting clients during training or performing model aggregation.

D. Data Cleaning and Handling False Data Injection

In a real-world FL-based IoT environment, the IoT clients generate their models based on their extracted data. In a conventional FL-based IoT approach, there is no intermediate stage to refine the sensor data, which may cause a falsified local model with an erroneous update that eventually misleads the global model aggregation. As the number of such false data injected clients increases, the model accuracy reduces at the same phase. In time, it brings down the chance of reaching convergence. Bagdasaryan *et al.* [60] proposed a backdoor FL to identify malicious attacks during federated aggregation. They developed a train-and-scale scheme to restrict anomaly detectors from looking at the client's model weights or accuracy during FL tasks. Fung *et al.* [176] explained the vulnerability of sybil attacks in the FL process. They proposed a defense mechanism that can identify poisoning sybils by analyzing the diversity of FL clients during model training. However, none of them considered real-time false-data injection onto the IoT clients, which needs further research.

E. Reducing Energy Consumption and On-Device Training

In line with our previous discussion, the clients of the IoT domain may possess a weak-processing unit. Therefore, it is challenging to conduct inference, training on devices, and executing timely interaction with the server through an energy-efficient communication scheme. However, on-device training causes two problems. First, the generated on-device model size needs to be small enough so that it fits within the device memory and still captures most of the data complexity to compute an effective model. The on-device inference problems are solved in [177] and [178], but the on-device training issues are not expounded. Second, the system can require high computational and storage availability for on-device training than these IoT clients can provide. Section IV-E presented some approaches suitable for specialized neumorphic or field-programmable gate array (FPGA)

TABLE V
ASYNCHRONOUS FL-BASED APPROACHES

Approaches	Ref.	Key ideas
ASO-fed	[166]	Edge devices train themselves through online learning and server performs aggregation by dynamic learning step size and exponential moving average.
TWAFL	[167]	Apply synchronous learning technique on client participants and temporally weighted aggregation on the local client models.
FedAsync	[97]	Optimize a scheduler for selecting client for training phase periodically.
Blockchain Asynch	[168]	Designed a blockchain-based asynchronous FL communication in internet of vehicles
FedAsynch Geo	[160]	Applied the federated asynchronous approach in geospatial applications and analyzed the performance and difference with synchronous aggregation technique.
FedAsynch DP	[162]	Adapted differential privacy for a secure and robust asynchronous FL based scheme.
SAFA	[163]	Semi-asynchronous FL to avoid problems regarding synchronous and asynchronous FL for attaining less communication overhead.

hardware or miss the combative constraints observed in the FL-IoT domain. Figuring out the solution to this dual problem is paramount. A potential direction can be found in [179]. They proposed an IoT-based network architecture to enable creating high-capacity client models with 15-38x fewer parameters compared to the conventional model experienced for such applications. Kumar *et al.* [177] proposed a tree-based approach to predict 2-kB RAM IoT devices, e.g., Arduino Uno board that possessed 8-bit ATmega328P microcontroller without any floating-point support, and 32 KB size of the read-only flash. Their proposed algorithm attains standard prediction accuracy by constructing a tree model that shrinks the model size and reduces prediction costs. They learned a sparse tree with high-powered nodes, carrying out the tree's learning process through sparsely projecting data within a low-dimensional space, and collaboratively learning all projection parameters and trees. Investigating such architecture to enable learning within the resource-constrained FL environments is an unexplored domain.

As we discussed, FL needs on-device training; therefore, any research that can enable energy-efficient execution of ML algorithms can help FL in turn. If we consider resource-constrained nodes for an IoT environment, then prolonging battery-power life duration is a challenge. Due to repeated interactions with the server, the battery charge can be reduced significantly. Minimizing the depreciation of battery power while interacting with the server is challenging. Kumar *et al.* [177] developed a tree-based algorithm for making a prediction on resource-constrained IoT devices (i.e., Arduino) that possess only 2-kB RAM. Still, they did not perform the training operation on resource-constrained IoT nodes. Gupta *et al.* [180] designed a kNN-based algorithm that works on resource-scarce IoT nodes ($\leq$ 32-kB RAM and 16-MHz processor) to predict through supervised learning, but the edge devices are not trained locally. Therefore, it is essential to design an improved version of the FL algorithm that can handle small computational power as well as storage to train IoT nodes on edges, and how we can manage the energy consumption of client nodes during the training phase is also an open issue. An exciting direction in this front is dynamic computation technologies. Dynamic computation techniques activate only a part of the NN for an input. This can help achieve both efficient training and inference as only a part of the NN is updated for each input. There are many different ways one can introduce dynamic computation to a NN. The techniques can be divided into three broad categories—dynamic channel pruning (DCP), dynamic-layer skipping (DLS), dynamic spatial gating (DSG), DLS, and Mixture of Experts (MoE). DSG techniques ([181], [182]) identify spatial regions in the OFM that are deemed important and focus their attention only on those parts of the OFM. DCP techniques identify channels in the OFM that are deemed unimportant and skip computations for those channels ([183], [184]). DLS techniques are more specific to ResNet style architecture with skip connections or RNN-based models ([185], [186]). Finally, MoE is based on the idea that instead of using a single large NN to process an input data, multiple domain experts can be used to process the input. Based on the input, the results from the domain experts can be given more weight. The routing to the experts is done via a predictor network. By gating experts that are deemed unimportant for the input, one can achieve faster computation [187]. DCP techniques can be viewed as a specific instantiation of the techniques in this domain.

Beyond algorithm innovations, there has been a surge of work in the domain of design of software and hardware that executes ML algorithms efficiently. Here, efficiency refers to any or all of reduced energy consumption, faster runtime, and smaller memory footprint. The works in this area can be categorized in the domain of novel instructions for executing ML in the CPU [188], [189], design of specialized accelerators [190]–[192], optimized software library [193], [194], development of new memory technologies [195], and near data processing to enable large storage using smaller energy budget [196]. However, the vast majority of these works are focused on the efficient inference of ML algorithms. A lot of these optimizations could be tailored to enable suitable training. Thus, further research in understanding the training algorithms of ML and how they execute on hardware can help tailor these solutions to solve this issue. The work described above modifies traditional hardware to make them amenable to ML. In traditional hardware, the unit responsible for processing information (processing unit) is separated from the unit responsible for storing data (memory). The instructions and data are fetched from memory and executed in the processing unit. This is called the von Neumann architecture. Apart from this, there is an entire body of works in the domain of neuromorphic computing [197], [198] dedicated to replicating the extreme power efficiency of the human brain by developing

new hardware that mimics its synaptic structure. The main difference with traditional hardware lies in the non von Neumann architecture of these hardware as the processing and memory elements are not separate. We refer the readers to [197] to get a better overview of this field. FL can also benefit from edge units built using this neuromorphic hardware that can enable efficient on-device learning.

F. Managing Dropped Participants

Internet availability and network connection power are crucial, particularly while applying FL in an IoT environment. Any FL-IoT participant may go out of the network in the middle of the training phase or during the interaction with the server due to mobility, bandwidth shortage, lack of transmission power, or out of battery life. Most of the recent works considered that all FL participants maintain a continuous connection with the server and cannot drop connection in the middle. In the real-world FL-IoT environment, such a scenario is not feasible, and any participant may go offline due to out-of-resources. Dropping off a significant portion of the participants would fail to generate an effective global model. It is difficult to understand whether a client gives a slow response because of the network issue or resource-shortage. Figuring out the potential problem may help us act according to the problem scenario. Das and Brunschwiler [92] presented a solution to handle straggler clients by acknowledging their resource utilization (i.e., computation power) after each local update. They formed a predictive model by analyzing the client's resource utilization and adjusting local computation accordingly. Another strategy is to perform asynchronous training, i.e., updating the global model whenever it receives a model update from any of the participated clients [67], [165]. Moreover, a recently invented FL framework, FedProx [199] can handle heterogeneity in federated networks. FedProx allows a partial amount of work from each client device through a reparameterization of the conventional FedAvg algorithm. However, when most of the clients within the network perform a low amount of partial works, their approach may take longer to reach convergence. Imteaj and Amini [126] proposed the activity and resource-aware FL strategy that can handle straggler issues by examining resource status, labeling clients with trust values in accordance to their contributions toward model convergence, and accepting variable works from the participated clients. However, further research needs to be conducted to optimize hyperparameters while enabling variable or partial works from the clients.

G. Privacy Preservation

The existing FL approaches aim to improve privacy by adapting classical cryptographic protocols and algorithms, such as DP and SMC. An FL-based SMC protocol is proposed in [73] to protect client model updates. Through this method, the server cannot see the local update parameter but can still extract some information by observing the aggregated results after each round. However, this approach encounters a high communication cost, which is not feasible for a resource-constrained FL-IoT environment. Geyer *et al.* [40], McMahan *et al.* [56], and Zhao *et al.* [200] applied DP to FL to achieve a global DP, but the hyperparameters of these approaches affect the communication and model accuracy. An adaptive gradient clipping technique is presented in [201] to handle this issue. In [58], a modified version of local privacy is designed to limit the power of adversaries that guarantees more robust privacy than global privacy and results in better model accuracy. Another interesting approach to a DP mechanism based on metalearning is proposed in [136] that can be used in FL through personalization. Besides, DP can be coupled with model compression strategies to reduce communication overhead and attain an improved version of privacy simultaneously [7], [135]. Furthermore, some prior works [202]–[205] proposed mechanisms to preserve privacy in a blockchain-enabled FL-based IoT environment. However, most of these approaches did not consider the heterogeneous resources of clients. They did not analyze the feasibility of applying a robust privacy-preserving algorithm that can be adapted without a straggler effect. Further research needs to be conducted to obtain maximum privacy benefits for the resource-constrained heterogeneous FL-based IoT environment.

VI. Applications of Federated Learning

FL fits best in applications where we need to deal with sensitive information and, therefore, on-device training is more important than passing local data to the server. Most of the existing FL applications are based on labeled data collected from clients or user activities (e.g., type URLs or keyboard, click button). In this section, we discuss some existing FL applications to better understand the real-world impact of FL.

A. Resource-Sufficient Federated Learning Application

Recommendation System: A recommendation system can be compared to an information filtering scheme that tries to predict user preference or rating for an item. In the conventional recommendation system, user preference or rating would be shared with other users, and privacy is not maintained in many cases. Instead of sharing such private data, Chen *et al.* [24] proposed a federated meta-learning framework through which each local client shares his/her algorithm rather than his/her data or local model. In particular, federated metalearning is useful when the model size is large; therefore, sharing the algorithm is more flexible than sharing a model.

Next-Word Prediction: A popular ML-based application is next-word prediction, where a model is constructed that can predict what the next probable word would be. Such a centralized ML application may transfer private user data (e.g., SMS and URLs) to the server and may leak any sensitive information about the user. From that motivation, an on-device distributed ML-based framework is designed by Hard *et al.* [22], which is inspired by the FedAvg algorithm. They trained each participant locally and obtained a higher recall than the conventional approach. In this way, FL helps a user make predictions by learning his/her typing behavior and indirectly reading the user's mind.

Keyword Spotting: Wake-word detector applications are prevalent nowadays. For instance, Amazon's "Hey Alexa"

wake-word detector is used to play different songs, or execute different commands, while Google's wake-word detector "Hey Google" is used for different purposes including driving, e.g., to get direction on a map. However, most of those applications are based on the cloud-based system and pass user data to the server. Unlike this, an embedded speech model is proposed [23], where they used a wake-word detector "Hey Snips" to recognize the user's voice. They used a crowdsourced data set and applied the FL strategy by keeping user information private.

Relevant Content Suggestions for On-Device Keyboard: Google has recently implemented a virtual keyboard application, named Gboard, where they applied the FL strategy to suggest relevant content [29]. It works on user-click or ignores situations that are stored in training cache and the value is added when related contents are suggested. Based on user-click, the information is stored in the cache and fed into the on-device training process. In this work, inference and training are performed on-device. Only the model updated parameters are shared with the server, while globally trained models are deployed on each client.

B. Resource-Constrained Federated Learning Application

Smart Robotics: A lifelong reinforcement FL framework for mobile robots is proposed in [50]. They designed an architecture to enhance navigation systems of mobile robots to learn efficiently from prior knowledge and adapt to a new environment effectively. They used two types of transfer learning for fast adaptation of the mobile robots within a new environment. Their proposed system is scalable but lacks security, privacy, robustness, and sparsification.

Smart Object Detection: Yu and Liu [213] designed an approach of optimizing object detection by considering Kullback–Leibler divergence (KLD) during the measurement of weights divergence of the client's local models. They adapted the abnormal weight suppression technique to reduce the effects of weight divergence that may be caused by unbalanced and non-IID data.

Smart Healthcare: In healthcare services, the FL-based IoT concept can be extremely effective to preserve the privacy of sensitive medical data. The IoT devices can be useful to generate data streams of patient's status, and FL can be used to undertake early precautions or treatment utilizing the historical data. Yuan *et al.* [214] developed an FL framework for smart healthcare by applying the FL mechanism and reduced computation load of IoT devices during training. Their proposed approach also took the edge of communication overhead during interaction of FL server and IoT devices. However, their developed framework does not guarantee convergence, and is incapable of performing a successful learning process in the presence of malfunction or edge/cloud server failure.

On-Device Ranking: Another application of FL is to rank a search result. For instance, if we query something in our device, an automated search result appeared. This is done by making an expensive call to the server. To reduce such cost, implementation of on-device training to generate a ranking of search results is proposed in [73], which is particularly useful for resource-constrained devices. By observing the user's selected item from a ranked list, their system puts a label whenever a user interacts with the ranking feature. In this way, user preference is not revealed to anyone, and communication overhead is reduced by a significant margin.

Anomaly Detection: An autonomous self-learning scheme is proposed in [215] to identify compromised devices within IoT networks. Relying on unlabeled crowdsource data and depending on the device-type-specific behavior profiles, their proposed system can learn the anomaly detection model without requiring any labeled data or human intervention to operate. They apply the FL strategy to aggregate behavior profiles for effective intrusion detection.

Resource-Efficient Training of UAV-Enabled IoT Devices: A particle swarm-based air quality monitoring framework is proposed in [57]. Their proposed system enables energy-efficient lightweight model training of unmanned aerial vehicles (UAVs) using aerial haze images and predicts air quality index (AQI) while preserving privacy. To sense ground systems, they proposed a Graph CNN-based long short-term memory (LSTM) model for obtaining accurate and real-time AQI inference. Besides, Tang *et al.* [216] addressed the issue of reducing latency and improving the energy efficiency of UAV-enabled IoT devices by optimizing battery resources and wireless bandwidth. They employed a deep deterministic policy gradient (DDPG) strategy to evaluate their system cost.

In Table VI, we present a detailed summary of some existing FL applications.

VII. Future Directions for Federated Learning Algorithms Considering Resource-Constrained IoT Devices

As we discussed, FL is a recently invented distributed ML technique that can be considered as an emerging research area. After examining the core challenges of the FL process while applying on resource-constrained IoT devices in Section IV, and analyzing some potential solutions of those emerging challenging in Section V, we point-out potential future directions in FL-based IoT environments. In this section, we highlight the future directions of this domain.

1) In an FL-based IoT environment, it can be experienced that some clients possess more data (e.g., due to frequent use of a particular application or having a greater memory size) compared to other clients within the underlying IoT network setting. Such a discrepancy in the number of data samples, particularly due to heterogeneous memory size and availability, leads to massive deviation in terms of training periods from participant-to-participant. The nonuniform data distribution raises issues in generating the representation of the population distribution of any client data set. Handling such disparity within the *local training data set* needs further research.
2) To ensure convergence for asynchronous learning in a Non-IID setting in the presence of resource-constrained

TABLE VI
LIST OF EXISTING FL APPLICATIONS

Approach	Category	Data partition	Model type	Decision-making	Remark
MDLdroid [209]	Applications	Horizontal	NN	Server-centric	Reinforcement learning
FedNER [210]					
FedRec [211]					
In-edge AI [11]					
FCF [212]			LM		Collaborative filter
FedMF [213]					Matrix factorization
FL Next-Word Prediction [22]			NN		Natural language processing
FL Emoji-Prediction [69]					
FL OOV [214]					
Gboard [29]			LM		Linear regression
FL RBHS [215]			LM		SVM

devices, loss functions of the *nonconvex problem* (i.e., an objective function that has multiple feasible regions, and each region has multiple locally optimal points) need to be considered, and supportive algorithms should be proposed.

3) In an FLS, we may need to choose a cluster head that would be responsible for passing the aggregated model parameter to the server for *energy efficiency*. The cluster head can collect client model parameters from its region in a synchronized fashion, while the central server can receive those locally aggregated models through a synchronous or asynchronous manner. Here, the leader can act as an intermediate aggregator and can avoid straggler nodes. It can marginally reduce power consumption as well as can minimize bandwidth requirements of the FLS, which could be effective for resource-constrained FL-based IoT environment. However, the leader node needs to be proficient to conduct swift operations and should be trustworthy to avoid false data injection.
4) A *device-centric automatic wake-up mechanism* can be useful in determining the optimal period to carry out interaction with the server. Such an approach can reduce unnecessary communication with the server and avoid sending a model update when the client's local data does not change much. Besides, the automatic wake-up mechanism may help the resource-constrained devices to reserve energy, which could be utilized in further training.
5) *Client mobility* can drastically change the overall system behavior. A network may hold a large number of active clients before the training starts, and after some period, most of them could go out of network. As a result, some areas may own a large number of clients, while some other regions may not be able to generate a feasible model due to a lack of active clients. Therefore, how to handle the mobility issues of the IoT devices and ensure successful federated model training is a potential research direction.
6) During the FL process, we may observe *statistical heterogeneity* among the client data. Such heterogeneity compels the clients to perform more interactions with the server, or with its neighbor nodes. The existing works did quantification of such statistical heterogeneity after initiating the training, which may cost extra resources and may have crucial impact on local training of the resource-constrained devices. Extensive research needs to be conducted to quantify the statistical heterogeneity even before the initialization of FL training to avoid idiosyncratic situations due to data sample variations.
7) *Effective incentives mechanism design* is essential to encourage FL clients to share their model information. Some incentives or regulations schemes are implemented in blockchain [217]–[219], and some incentive mechanisms are proposed for high-quality federated data [220], [221]. Still, more extensive research needs to be conducted on the incentives mechanism design to upgrade the effectiveness of FL. An example of such a design is how game-theory models can be adapted in FLS or, in addition to the accuracy, what new benefit can be provided to the user to encourage them for joining in FL training. Besides, as the FL participants can be resource-bounded, or the participants can be business competitors, it is mandatory to design a strategy that divides the overall earnings to ensure the long-term engagement of the participants. Furthermore, more focus needs to be placed on how to defend against adversarial attacks that try to collect the majority of the incentives.
8) A *lightweight blockchain* framework for the FL-IoT setting needs to be designed that can ensure robustness and enhance privacy and security while interacting with the server or neighbor clients. Blockchain can prevent model parameters or algorithm temperament and verify the model update and exchange. Some blockchain paradigm for on-device training is discussed in [36] and [222], but they designed that the framework without considering the challenges of the weak-processing unit and limited memory of IoT devices. Further research needs to be conducted on designing miner selection, block mining, consensus algorithm, validating a chain, atomicity, and blockchain interoperability, especially for FL with resource-constrained IoT clients.

9) The FL structure leads us to think about integrating *trust model* to avoid adversarial clients during training. Selecting a client based on only resource availability would lead us to choose a malicious client. However, we can design a trust-based model based on the client's previous contribution to learning within the network and interacting with other clients. Typically, it is assumed that the server is trustworthy, and we can use the server to generate the trust model by analyzing the behavior of the clients. The incentive mechanism can be designed based on the generated trust model, and this may open us a new research direction.

VIII. Conclusion

This article presented a comprehensive survey on FL algorithms and analyzed the implementation challenges while performing on-device training. We particularly emphasized the issues of the FL process while considering resource-constrained IoT devices as FL clients. First, we presented a highlight over FL algorithms that can enable efficient and scalable model training in edge devices. Then, we presented an overview of the FL taxonomy and analyzed the existing papers to distinguish our contribution as compared to prior surveys. We discussed distributed learning and optimization techniques and explained various aspects of distributed algorithms for decision-making purposes. We analyzed the challenges during the on-device learning of resource-constrained devices and discussed existing feasible solutions. After analyzing the challenges, we described the emerging challenges of FL implementation in resource-constrained IoT devices, which needs extensive further research. Afterward, we explored the existing FL applications to provide a better understanding of the FL role in real-world applications. Finally, we listed the potential future directions of deploying FL within the resource-constrained heterogeneous IoT environment.

References

[1] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*, Chicago, IL, USA, 1982, pp. 160–164.

[2] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 439–450.

[3] R. J. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, Tokyo, Japan, 2005, pp. 217–228.

[4] J. Vaidya, H. Yu, and X. Jiang, "Privacy-preserving SVM classification," *Knowl. Inf. Syst.*, vol. 14, no. 2, pp. 161–178, 2008.

[5] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, "Protecting data privacy in private information retrieval schemes," *J. Comput. Syst. Sci.*, vol. 60, no. 3, pp. 592–629, 2000.

[6] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-preserving data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM 26th IEEE Int. Conf. Comput. Commun.*, Anchorage, AK, USA, 2007, pp. 2045–2053.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[8] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 10, no. 2, p. 12, 2019.

[9] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities and challenges," 2019. [Online]. Available: arXiv:1908.06847.

[10] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[11] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep./Oct. 2019.

[12] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[13] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[14] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[15] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for Internet of Things," *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 8, pp. 1399–1417, 2018.

[16] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[17] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[18] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[19] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[20] H B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016. [Online]. Available: arXiv:1602.05629.

[21] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018. [Online]. Available: arXiv:1812.06127.

[22] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," 2018. [Online]. Available: arXiv:1811.03604.

[23] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Brighton, U.K., 2019, pp. 6341–6345.

[24] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," 2018. [Online]. Available: arXiv:1802.07876.

[25] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "SecureBoost: A lossless federated learning framework," 2019. [Online]. Available: arXiv:1901.08755.

[26] Y. Liu *et al.*, "A communication efficient vertical federated learning framework," 2019. [Online]. Available: arXiv:1912.11187.

[27] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[28] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," 2018. [Online]. Available: arXiv:1812.03337.

[29] T. Yang *et al.*, "Applied federated learning: Improving Google keyboard query suggestions," 2018. [Online]. Available: arXiv:1812.02903.

[30] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," 2019. [Online]. Available: arXiv:1911.04206.

[31] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017. [Online]. Available: arXiv:1711.10677.

[32] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. IEEE Symp. Security Privacy*, Berkeley, CA, USA, 2013, pp. 334–348.

[33] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016. [Online]. Available: arXiv:1610.05492.

[34] H. Wang, M. Yurochkin, Y. Sun, D. S. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020. [Online]. Available: arXiv:2002.06440.

[35] Q. Li, Z. Wen, and B. He, "Federated learning systems: Vision, hype and reality for data privacy and protection," 2019. [Online]. Available: arXiv:1907.09693.

[36] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[37] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Security*, 2019, pp. 1–11.

[38] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," 2019. [Online]. Available: arXiv:1902.00146.

[39] L. Zhao *et al.*, "InPrivate digging: Enabling tree-based distributed data mining with differential privacy," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 2087–2095.

[40] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017. [Online]. Available: arXiv:1712.07557.

[41] A. Agarwal, J. Langford, and C.-Y. Wei, "Federated residual learning," 2020. [Online]. Available: arXiv:2003.12880.

[42] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," 2015. [Online]. Available: arXiv:1511.03575.

[43] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, T. N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," 2019. [Online]. Available: arXiv:1905.12022.

[44] Y. Liu, Z. Ma, X. Liu, S. Ma, S. Nepal, and R. H. Deng, "Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing," 2019. [Online]. Available: arXiv:1907.10218.

[45] Y. Liu, Y. Liu, Z. Liu, J. Zhang, C. Meng, and Y. Zheng, "Federated forest," 2019. [Online]. Available: arXiv:1905.10053.

[46] Y.-R. Chen, A. Rezapour, and W.-G. Tzeng, "Privacy-preserving ridge regression on distributed data," *Inf. Sci.*, vols. 451–452, pp. 34–49, Jul. 2018.

[47] A. P. Sanil, A. F. Karr, X. Lin, and J. P. Reiter, "Privacy preserving regression modelling via distributed computation," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Disco. Data Min.*, 2004, pp. 677–682.

[48] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2017, pp. 4424–4434.

[49] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," 2019. [Online]. Available: arXiv:1909.12488.

[50] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4555–4562, Oct. 2019.

[51] C. He, M. Annavaram, and S. Avestimehr, "FedNAS: Federated deep learning via neural architecture search," 2020. [Online]. Available: arXiv:2004.08546.

[52] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1310–1322, Apr. 2020.

[53] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," 2018. [Online]. Available: arXiv:1811.11479.

[54] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[55] R. Liu, Y. Cao, M. Yoshikawa, and H. Chen, "FedSel: Federated SGD under local differential privacy with top-k dimension selection," 2020. [Online]. Available: arXiv:2003.10637.

[56] H B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," 2017. [Online]. Available: arXiv:1710.06963.

[57] Y. Liu, J. Nie, X. Li, S. H. Ahmed, W. Y. B. Lim, and C. Miao, "Federated learning in the sky: Aerial-ground air quality sensing framework with UAV swarms," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9827–9837, Jun. 2021.

[58] A. Bhowmick, J. C. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2018. [Online]. Available: arXiv:1812.00984.

[59] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 1175–1191.

[60] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," 2018. [Online]. Available: arXiv:1807.00459.

[61] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. B. Calo, "Analyzing federated learning through an adversarial lens," 2018. [Online]. Available: arXiv:1811.12470.

[62] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[63] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," 2019. [Online]. Available: arXiv:1905.10497.

[64] Y. Liu, S. Sun, Z. Ai, S. Zhang, Z. Liu, and H. Yu, *FedCoin: A Peer-to-Peer Payment System for Federated Learning*. Cham, Switzerland: Springer, 2020.

[65] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[66] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, 2019, pp. 1–7.

[67] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[68] S. Caldas *et al.*, "LEAF: A benchmark for federated settings," 2018. [Online]. Available: arXiv:1812.01097.

[69] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," 2019. [Online]. Available: arXiv:1906.04329.

[70] A. Khaled, K. Mishchenko, and P. Richtárik, "First analysis of local GD on heterogeneous data," 2019. [Online]. Available: arXiv:1909.04746.

[71] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized collaborative learning of personalized models over networks," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, 2017, pp. 509–517.

[72] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2017, pp. 5330–5340.

[73] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019. [Online]. Available: arXiv:1902.01046.

[74] M. Assran, N. Loizou, N. Ballas, and M. G. Rabbat, "Stochastic gradient push for distributed deep learning," 2018. [Online]. Available: arXiv:1811.10792.

[75] I. Colin, A. Bellet, J. Salmon, and S. Clémençon, "Gossip dual averaging for decentralized optimization of pairwise functions," 2016. [Online]. Available: arXiv:1606.02421.

[76] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: Decentralized training over decentralized data," in *Proc. Int. Conf. Mach. Learn.*, 2018. pp. 4848–4856.

[77] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," 2019. [Online]. Available: arXiv:1902.00340.

[78] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," 2017. [Online]. Available: arXiv:1705.08435.

[79] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "GADMM: Fast and communication efficient framework for distributed machine learning," 2019. [Online]. Available: arXiv:1909.00047.

[80] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," 2019. [Online]. Available: arXiv:1901.11173.

[81] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "MLbase: A distributed machine-learning system," in *Proc. Biennial Conf. Innovat. Data Syst. Res. (CIDR)*, vol. 1, 2013, pp. 1–2.

[82] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2014, pp. 19–27.

[83] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project, Zug, Switzerland, Yellow Paper, 2014.

[84] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1000–1008.

[85] M. Jaggi *et al.*, "Communication-efficient distributed dual coordinate ascent," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2014, pp. 3068–3076.

[86] Y. Zhang and L. Xiao, "DiSCO: Distributed optimization for self-concordant empirical loss," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 362–370.

[87] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2013, pp. 315–323.

[88] J. Konečnỳ and P. Richtárik, "Semi-stochastic gradient descent methods," 2013. [Online]. Available: arXiv:1312.1666.

[89] S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup, and M. Shah, "On-device machine learning: An algorithms and learning theory perspective," 2019. [Online]. Available: arXiv:1911.00623.

[90] C. Ma *et al.*, "Distributed optimization with arbitrary local solvers," *Optim. Methods Softw.*, vol. 32, no. 4, pp. 813–848, 2017.

[91] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. R. Cadambe, "Trading redundancy for communication: Speeding up distributed SGD for non-convex optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 2545–2554.

[92] A. Das and T. Brunschwiler, "Privacy is what we care about: Experimental investigation of federated learning on edge devices," 2019. [Online]. Available: arXiv:1911.04559.

[93] Y. Jiang, S. Wang, B.-J. Ko, W.-H. Lee, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," 2019. [Online]. Available: arXiv:1909.12326.

[94] Z. Xu, Z. Yang, J. Xiong, J. Yang, and X. Chen, "ELFISH: Resource-aware federated learning on heterogeneous edge devices," 2019. [Online]. Available: arXiv:1912.01684.

[95] M. Zinkevich, J. Langford, and A. J. Smola, "Slow learners are fast," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2009, pp. 2331–2339.

[96] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019. [Online]. Available: arXiv:1903.03934.

[97] S. Zheng *et al.*, "Asynchronous stochastic gradient descent with delay compensation," in *Proc. 34th Int. Conf. Mach. Learn. Vol. 70*, 2017, pp. 4120–4129.

[98] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," 2017. [Online]. Available: arXiv:1710.06952.

[99] Z. Chai *et al.*, "Towards taming the resource and data heterogeneity in federated learning," in *Proc. USENIX Conf. Oper. Mach. Learn. (OpML)*, 2019, pp. 19–21.

[100] S. Ha, J. Zhang, O. Simeone, and J. Kang, "Coded federated computing in wireless networks with straggling devices and imperfect CSI," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, 2019, pp. 2649–2653.

[101] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "Federated learning for resource-constrained IoT devices: Panoramas and state-of-the-art," 2020. [Online]. Available: arXiv 2002.10610.

[102] A. Zhou, R. Muller, and J. M. Rabaey, "Memory-efficient, limb position-aware hand gesture recognition using hyperdimensional computing," 2021. [Online]. Available: arXiv:2103.05267.

[103] C. R. Banbury *et al.*, "Benchmarking tinyml systems: Challenges and direction," 2021. [Online]. Available: arXiv:2003.04821.

[104] C. Banbury *et al.*, "Microets: Neural network architectures for deploying TinyML applications on commodity microcontrollers," 2021. [Online]. Available: arXiv:2010.11267.

[105] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[106] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," 2018. [Online]. Available: arXiv:1802.04680.

[107] J. Park *et al.*, "Distilling on-device intelligence at the network edge," 2019. [Online]. Available: arXiv:1908.05895.

[108] N. S. Sohoni, C. R. Aberger, M. Leszczynski, J. Zhang, and C. Ré, "Low-memory neural network training: A technical report," 2019. [Online]. Available: arXiv:1904.10631.

[109] P. Jain *et al.*, "Checkmate: Breaking the memory wall with optimal tensor rematerialization," in *Proc. Int. Conf. Mach. Learn. Syst.*, vol. 2, 2020, pp. 497–511.

[110] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," 2016. [Online]. Available: arXiv:1604.06174.

[111] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," 2021. [Online]. Available: arXiv:2007.11622.

[112] Z. Gu *et al.*, "Reaching data confidentiality and model accountability on the caltrain," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, Portland, OR, USA, 2019, pp. 336–348.

[113] T. Tuor, S. Wang, B. J. Ko, C. Liu, and K. K. Leung, "Overcoming noisy and irrelevant data in federated learning," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, 2020, pp. 5020–5027.

[114] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," 2019. [Online]. Available: arXiv:1909.11875.

[115] L. Lyu, X. Xu, and Q. Wang, "Collaborative fairness in federated learning," 2020. [Online]. Available: arXiv:2008.12161.

[116] H. Yu *et al.*, "A fairness-aware incentive scheme for federated learning," in *Proc. AAAI/ACM Conf. AI Ethics Soc.*, 2020, pp. 393–399.

[117] L. U. Khan *et al.*, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, Oct. 2020.

[118] Y. Liu and J. Wei, "Incentives for federated learning: A hypothesis elicitation approach," 2020. [Online]. Available: arXiv:2007.10596.

[119] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "FLChain: A blockchain for auditable federated learning with trust and incentive," in *Proc. 5th Int. Conf. Big Data Comput. Commun. (BIGCOM)*, QingDao, China, 2019, pp. 151–159.

[120] T. Nishio, R. Shinkuma, and N. B. Mandayam, "Estimation of individual device contributions for incentivizing federated learning," 2020. [Online]. Available: arXiv:2009.09371.

[121] M. Cong, H. Yu, X. Weng, J. Qu, Y. Liu, and S.-M. Yiu, "A VCG-based fair incentive mechanism for federated learning," 2020. [Online]. Available: arXiv:2008.06680.

[122] H. Yu *et al.*, "A sustainable incentive scheme for federated learning," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 58–69, Jul./Aug. 2020.

[123] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," 2019. [Online]. Available: arXiv:1909.07972.

[124] A. Imteaj and M. H. Amini, "FedAR: Activity and resource-aware federated learning model for distributed mobile robots," in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Miami, FL, USA, Dec. 2020, pp. 1153–1160.

[125] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, 2020.

[126] A. Imteaj and M. H. Amini, "FedPARL: Client activity and resource-oriented lightweight federated learning model for resource-constrained heterogeneous IoT environment," *Front. Commun. Netw.*, vol. 2, no. 10, 2021, Art. no. 657653.

[127] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, 2019, pp. 691–706.

[128] Ni. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: Measuring unintended neural network memorization & extracting secrets," 2018. [Online]. Available: arXiv:1802.08232.

[129] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1322–1333.

[130] M S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Security*, 2018, pp. 707–721.

[131] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *J. Cryptol.*, vol. 1, no. 1, pp. 65–75, 1988.

[132] N. Guha, A. Talwlkar, and V. Smith, "One-shot federated learning," 2019. [Online]. Available: arXiv:1902.11175.

[133] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6532–6542, Oct. 2020.

[134] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, 2019, pp. 739–753.

[135] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2018, pp. 7564–7575.

[136] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," 2019. [Online]. Available: arXiv:1909.05830.

[137] W. Li *et al.*, "Privacy-preserving federated brain tumour segmentation," in *Proc. Int. Workshop Mach. Learn. Med. Imag.*, 2019, pp. 133–141.

[138] B. Ghazi, R. Pagh, and A. Velingker, "Scalable and differentially private distributed aggregation in the shuffled model," 2019. [Online]. Available: arXiv:1906.08320.

[139] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Paris, France, 2019, pp. 2512–2520.

[140] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No peek: A survey of private distributed deep learning," 2018. [Online]. Available: arXiv:1812.03288.

[141] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

[142] J. Zhang, B. Chen, S. Yu, and H. Deng, "PEFL: A privacy-enhanced federated learning scheme for big data analytics," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1–6.

[143] M. Seif, R. Tandon, and M. Li, "Wireless federated learning with local differential privacy," 2020. [Online]. Available: arXiv:2002.05151.

[144] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10782–10793, Nov. 2020.

[145] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li, "A training-integrity privacy-preserving federated learning scheme with trusted execution environment," *Inf. Sci.*, vol. 522, pp. 69–79, Jun. 2020.

[146] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Edge-assisted hierarchical federated learning with non-IID data," 2019. [Online]. Available: arXiv:1905.06641.

[147] A. Imteaj and M. H. Amini, "Distributed sensing using smart end-user devices: Pathway to federated learning for autonomous IoT," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vegas, NV, USA, 2019, pp. 1156–1161.

[148] Y. Sun, S. Zhou, and D. Gündüz, "Energy-aware analog aggregation for federated learning with redundant data," 2019. [Online]. Available: arXiv:1911.00188.

[149] P. Han, S. Wang, and K. K. Leung. "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Singapore, 2020, pp. 300–310.

[150] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018. [Online]. Available: arXiv:1812.07210.

[151] H. Sun, S. Li, F. R. Yu, Q. Qi, J. Wang, and J. Liao, "Toward communication-efficient federated learning in the Internet of Things with edge computing," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 11053–11067, Nov. 2020.

[152] Y. Liu *et al.*, "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348–6358, Apr. 2021.

[153] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with Moreau envelopes," 2020. [Online]. Available: arXiv:2006.08848.

[154] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning and permissioned blockchain for digital twin edge networks," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2276–2288, Feb. 2021.

[155] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, "Personalized federated learning: A meta-learning approach," 2020. [Online]. Available: arXiv:2002.07948.

[156] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," 2019. [Online]. Available: arXiv:1911.02417.

[157] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, Mar. 2020.

[158] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Paris, France, 2019, pp. 1387–1395.

[159] M. R. Sprague *et al.*, "Asynchronous federated learning for geospatial applications," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disco. Databases (ECML-PKDD)*, vol. 967. Dublin, Ireland, 2018, pp. 21–28.

[160] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2134–2143, Mar. 2020.

[161] W. Wu, L. He, W. Lin, R. Mao, and S. A. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," 2019. [Online]. Available: arXiv:1910.01355.

[162] Y. Li, S. Yang, X. Ren, and C. Zhao, "Asynchronous federated learning with differential privacy for edge intelligence," 2019. [Online]. Available: arXiv:1912.07902.

[163] Y. Zhang, Y. Lu, X. Huang, K. Zhang, and S. Maharjan, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298–4311, Apr. 2020.

[164] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices," 2019. [Online]. Available: arXiv:1911.02134.

[165] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.

[166] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.

[167] S. Thrun and L. Pratt, *Learning to Learn*. New York, NY, USA: Springer, 2012.

[168] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[169] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Disco. Data Min.*, 2004, pp. 109–117.

[170] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018. [Online]. Available: arXiv:1806.00582.

[171] L. Corinzia and J. M. Buhmann, "Variational federated multi-task learning," 2019. [Online]. Available: arXiv:1906.06268.

[172] H. Eichner, T. Koren, H. B. McMahan, N. Srebro, and K. Talwar, "Semi-cyclic stochastic gradient descent," 2019. [Online]. Available: arXiv:1904.10120.

[173] M. Khodak, M.-F. Balcan, and A. Talwalkar, "Adaptive gradient-based meta-learning methods," 2019. [Online]. Available: arXiv:1906.02717.

[174] I. I. Eliazar and I. M. Sokolov, "Measuring statistical heterogeneity: The Pietra index," *Physica A, Stat. Mech. Appl.*, vol. 389, no. 1, pp. 117–125, 2010.

[175] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," 2019. [Online]. Available: arXiv:1910.03581.

[176] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018. [Online]. Available: arXiv:1808.04866.

[177] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 KB RAM for the Internet of Things," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1935–1944.

[178] U. Thakker *et al.*, "Compressing RNNs for IoT devices by 15-38x using kronecker products," 2019. [Online]. Available: arXiv:1906.02876.

[179] U. Thakker, J. G. Beu, D. Gope, G. Dasika, and M. Mattina, "Runtime efficient RNN compression for inference on edge devices," 2019. [Online]. Available: arXiv:1906.04886.

[180] C. Gupta *et al.*, "ProtoNN: Compressed and accurate kNN for resource-scarce devices," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1331–1340.

[181] W. Hua, Y. Zhou, C. De Sa, Z. Zhang, and G. E. Suh, "Channel gating neural networks," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2018.

[182] X. Huang, U. Thakker, D. Gope, and J. Beu, "Pushing the envelope of dynamic spatial gating technologies," in *Proc. 2nd Int. Workshop Challenges Artif. Intell. Mach. Learn. Internet Things*, 2020, pp. 21–26.

[183] R. Raju, D. Gope, U. Thakker, and J. Beu, "Understanding the impact of dynamic channel pruning on conditionally parameterized convolutions," in *Proc. 2nd Int. Workshop Challenges Artif. Intell. Mach. Learn. Internet Things*, 2020, pp. 27–33.

[184] X. Gao, Y. Zhao, L. Dudziak, R. D. Mullins, and C.-Z. Xu, "Dynamic channel pruning: Feature boosting and suppression," 2018. [Online]. Available: arXiv:1810.05331.

[185] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 420–436.

[186] J. Tao, U. Thakker, G. Dasika, and J. Beu, "Skipping RNN state updates without retraining the original model," in *Proc. 1st Workshop Mach. Learn. Edge Sens. Syst.*, 2019, pp. 31–36.

[187] N. Shazeer *et al.*, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017.

[188] S. Liu *et al.*, "Cambricon: An instruction set architecture for neural networks," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, 2016, pp. 393–405.

[189] N. Stephens *et al.*, "The ARM scalable vector extension," *IEEE Micro*, vol. 37, no. 2, pp. 26–39, Mar./Apr. 2017.
[190] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
[191] T. Chen *et al.*, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Comput. Archit. News*, vol. 42, no. 1, pp. 269–284, 2014.
[192] A. Samajdar, P. Mannan, K. Garg, and T. Krishna, "GeneSys: Enabling continuous learning through neural network evolution in hardware," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchit.*, 2018, pp. 855–866.
[193] U. Thakker, G. Dasika, J. G. Beu, and M. Mattina, "Measuring scheduling efficiency of RNNs for NLP applications," 2019. [Online]. Available: arXiv:1904.03302.
[194] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for arm cortex-M CPUs," 2018. [Online]. Available: arXiv:1801.06601.
[195] S. Koppula *et al.*, "EDEN: Enabling energy-efficient, high-performance deep neural network inference using approximate dram," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2019, pp. 166–181.
[196] M. Imani, M. Samragh, Y. Kim, S. Gupta, F. Koushanfar, and T. Rosing, "RAPIDNN: In-memory deep neural network acceleration framework," 2018. [Online]. Available: arXiv:1806.05794.
[197] C. D. Schuman *et al.*, "A survey of neuromorphic computing and neural networks in hardware," 2017. [Online]. Available: arXiv: 2011.00111.
[198] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, 2016, pp. 380–392.
[199] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Int. Conf. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.
[200] Y. Zhao *et al.*, "Local differential privacy-based federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, Jun. 2021.
[201] O. Thakkar, G. Andrew, and H. B. McMahan, "Differentially private learning with adaptive clipping," 2019. [Online]. Available: arXiv:1905.03871.
[202] Y. Qu *et al.*, "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5171–5183, Jun. 2020.
[203] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.
[204] Y. Zhao *et al.*, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb. 2021.
[205] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng, "Privacy-preserving federated learning framework based on chained secure multiparty computing," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6178–6186, Apr. 2021.
[206] Y. Zhang, T. Gu, and X. Zhang, "MDLdroid: A chainSGD-reduce approach to mobile deep learning for personal mobile sensing," in *Proc. 19th ACM/IEEE Int. Conf. Inf. Process. Sens. Netw.*, 2020, pp. 73–84.
[207] S. Ge, F. Wu, C. Wu, T. Qi, Y. Huang, and X. Xie, "FedNER: Privacy-preserving medical named entity recognition with federated learning," 2020. [Online]. Available: arXiv:2003.09288.
[208] T. Qi, F. Wu, C. Wu, Y. Huang, and X. Xie, "FedRec: Privacy-preserving news recommendation with federated learning," 2020. [Online]. Available: arXiv:2003.09592.
[209] M. Ammad-ud-din *et al.*, "Federated collaborative filtering for privacy-preserving personalized recommendation system," 2019. [Online]. Available: arXiv:1901.09888.
[210] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," 2019. [Online]. Available: arXiv:1906.05108.
[211] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, "Federated learning of out-of-vocabulary words," 2019. [Online]. Available: arXiv:1903.10635.
[212] F. Hartmann, S. Suh, A. Komarzewski, T. D. Smith, and I. Segall, "Federated learning for ranking browser history suggestions," 2019. [Online]. Available: arXiv:1911.11807.
[213] P. Yu and Y. Liu, "Federated object detection: Optimizing object detection model with federated learning," in *Proc. 3rd Int. Conf. Vis. Image Signal Process.*, 2019, pp. 1–6.
[214] B. Yuan, S. Ge, and W. Xing, "A federated learning framework for healthcare IoT devices," 2020. [Online]. Available: arXiv:2005.05083.
[215] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Dallas, TX, USA, 2019, pp. 756–767.
[216] S. Tang, W. Zhou, L. Chen, L. Lai, J. Xia, and L. Fan, "Battery-constrained federated edge learning in UAV-enabled IoT for B5G/6G networks," 2021. [Online]. Available: arXiv:2101.12472.
[217] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2016, pp. 45–59.
[218] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.
[219] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Security Privacy Workshops*, San Jose, CA, USA, 2015, pp. 180–184.
[220] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: A Stackelberg game perspective," *IEEE Netw. Lett.*, vol. 2, no. 1, pp. 23–27, Mar. 2020.
[221] R. Jurca and B. Faltings, "An incentive compatible reputation mechanism," in *Proc. EEE Int. Conf. E-Commerce*, Newport Beach, CA, USA, 2003, pp. 285–292.
[222] R. Xu, Y. Chen, and J. Li, "MicroFL: A lightweight, secure-by-design edge network fabric for decentralized IoT systems," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2020.

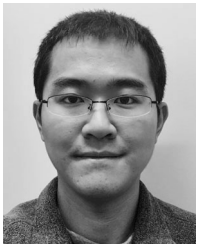
Ahmed Imteaj received the B.Sc. degree in computer science and engineering from Chittagong University of Engineering and Technology, Chittagong, Bangladesh, in 2015. He is currently pursuing the Ph.D. degree with Florida International University, Miami, FL, USA.

He is a Graduate Assistant with the Knight Foundation School of Computing and Information Sciences, Florida International University, where he is also a Research Lab Member with the Sustainability, Optimization, and Learning for Interdependent Networks Laboratory. From 2015 to 2018, he worked as a Lecturer with International Islamic University Chittagong, Chittagong. He has published more than 30 referred journals and conference papers. His research interests span federated learning, Internet of Things (IoT), machine learning, blockchain, sensor networks, cyber–physical–social resilience, and optimization.

Mr. Imteaj's work on federated learning for IoT environments was a recipient of the Best Paper Award from the "2019 IEEE Conference on Computational Science and Computational Intelligence" and won the Second Place at 2021 Florida International University GSAW Scholarly Forum. (Lab website: www.solidlab.network).

Urmish Thakker received the bachelor's degree from Birla Institute of Technology and Science Pilani, Pilani, India, in 2012, and the master's degree in computer science from UW Madison, Madison, WI, USA.

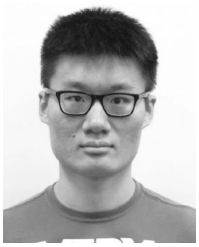
He is a Deep Learning Researcher with SambaNova Systems, Palo Alto, CA, USA. Before joining SambaNova, he worked with Arm Research, Texas Instruments and Broadcom, AMD, Santa Clara, CA, USA. Specifically, he has worked on model quantization, pruning, structured matrices and low-rank decomposition. His work has led to patents, publications, and contributions to various products across multiple companies. His research has primarily focused on efficient execution of neural networks on resource-constrained devices.



Shiqiang Wang (Member, IEEE) received the bachelor's and master's degrees from Northeastern University, Shenyang, China, in 2009 and 2011, respectively, and the Ph.D. degree from the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., in 2015.

He has been a Research Staff Member with IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, since 2016, where he was also a Graduate-Level Co-Op in the summers of 2014 and 2013. In the fall of 2012, he was with NEC Laboratories Europe, Heidelberg, Germany. His current research focuses on the interdisciplinary areas in distributed computing, machine learning, networking, optimization, and signal processing.

Dr. Wang received the IEEE Communications Society Leonard G. Abraham Prize in 2021, the IBM Outstanding Technical Achievement Award in 2019 and 2021, the multiple Invention Achievement Awards from IBM in 2016, the Best Paper Finalist of the IEEE International Conference on Image Processing 2019, and Best Student Paper Award of the Network and Information Sciences International Technology Alliance in 2015. He served as a Technical Program Committee Member of several international conferences, including ICML, NeurIPS, ICDCS, AISTATS, IJCAI, IFIP Networking, IEEE GLOBECOM, and IEEE ICC and an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING.



Jian Li (Member, IEEE) received the B.E. degree from Shanghai Jiao Tong University, Shanghai, China, in June 2012, and the Ph.D. degree in computer engineering from Texas A&M University at College Station, College Station, TX, USA, in December 2016.

He is an Assistant Professor of Computer Engineering with the Department of Electrical and Computer Engineering, State University of New York, Binghamton University, Binghamton, NY, USA. He was a Postdoctoral Fellow with the College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA, USA, from January 2017 to August 2019. His current research interests lie in the areas of reinforcement learning, online learning, network optimization, online algorithms, and their applications in large-scale networked systems.

M. Hadi Amini (Member, IEEE) received the B.Sc. degree from Sharif University of Technology, Tehran, Iran, in 2011, the M.Sc. degree from Tarbiat Modares University, Tehran, in 2013, the M.Sc. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2015 and 2019, respectively, and the Doctoral degree in computer science and technology from SYSU, Guangzhou, China, in 2018.

He is an Assistant Professor with the Knight Foundation School of Computing and Information Sciences, Florida International University, Miami, FL, USA. He is the Director of Sustainability, Optimization, and Learning for InterDependent Networks Laboratory (www.solidlab.network). He has published more than 100 refereed journal and conference papers, and book chapters. He has edited/authored six books. His research interests include distributed optimization and learning algorithms, distributed computing and intelligence, sensor networks, interdependent networks, and cyber–physical–social resilience. Application domains include smart cities, energy systems, transportation networks, and healthcare. (Homepage: www.hadiamini.com)

Dr. Amini was a recipient of the Best Paper Award from "2019 IEEE Conference on Computational Science and Computational Intelligence," FIU's Knight Foundation School of Computing and Information Sciences "Excellence in Teaching Award," Best Reviewer Award from Four IEEE Transactions, the Best Journal Paper Award in "Journal of Modern Power Systems and Clean Energy," and the Dean's Honorary Award from the President of Sharif University of Technology. He has served as the President of Carnegie Mellon University Energy Science and Innovation Club; as technical program committee of several IEEE and ACM conferences; and as the Lead Editor for a book series on *Sustainable Interdependent Networks: From Theory to Application* in 2017. He also serves as an Associate Editor for *SN Operations Research Forum*, *Frontiers in Communications and Networks* (Data Science for Communications), and *International Transactions on Electrical Energy Systems*. He is a Life Member of IEEE-Eta Kappa Nu (IEEE-HKN), the Honor Society of IEEE. (Lab website: www.solidlab.network).