Collaborative Machine Learning: Schemes, Robustness, and Privacy

Junbo Wang, *Member, IEEE*, Amitangshu Pal, *Member, IEEE*, Qinglin Yang, *Member, IEEE*, Krishna Kant, *Life Fellow, IEEE*, Kaiming Zhu, and Song Guo, *Fellow, IEEE*

***Abstract*— Distributed machine learning (ML) was originally introduced to solve a complex ML problem in a parallel way for more efficient usage of computation resources. In recent years, such learning has been extended to satisfy other objectives, namely, performing learning *in situ* on the training data at multiple locations and keeping the training datasets private while still allowing sharing of the model. However, these objectives have led to considerable research on the vulnerabilities of distributed learning both in terms of privacy concerns of the training data and the robustness of the learned overall model due to bad or maliciously crafted training data. This article provides a comprehensive survey of various privacy, security, and robustness issues in distributed ML.**

***Index Terms*— Collaborative learning, distributed learning, federated learning, privacy, robustness.**

NOMENCLATURE

RA	Root agent.
ML	Machine learning.
WA	Worker agent.
DNN	Deep neural network.
SVM	Support vector machine.
CNN	Convolutional neural network.
NN	Neural network.
GAN	Generative adversarial network.

I. INTRODUCTION

THE ML has been proven successful in numerous applications, including object recognition, autonomous driving, stock prediction, and so on. Collaborative ML provides a collaborative training framework, which expands the training process in multiple computing nodes to increase computing resources, avoid unnecessary transfer of large training datasets, and conduct training on datasets without having to reveal them. Collaborative ML algorithms can be based on a variety of learning schemes along with many ways of aggregating them.

While collaborative ML offers many advantages over centralized learning, it is also subject to various forms of attacks. In particular, the worker nodes may be malicious or untrustworthy, or their communications with the root node may be compromised by man-in-the-middle or other types of communication attacks. Also, the privacy of the data belonging to the worker node could be an issue [1]. Although these issues have been discussed in the literature under various assumptions [2]–[5], there is no integrated methodology to consider various forms of robustness challenges.

The purpose of this article is to address this gap. We describe several popular collaborative ML algorithms that use labeled data for training (e.g., collaborative clustering, collaborative SVM, collaborative decision tree (DT), and collaborative NN) and the suitable aggregation methods for them. In addition, unsupervised learning, which uses unlabeled data, can also be done in a distributed way but requires a somewhat different treatment. The discussion on different learning schemes provides a basis to discuss the robustness issue, where we provide a taxonomy of various types of attacks, including data poisoning attacks, model poisoning attacks, and black-, gray-, and white-box attacks. Then, we discuss and compare possible privacy leakage and the corresponding protection methods.

To the best of our knowledge, it is the first survey to discuss robustness and privacy leakage by involving various collaborative ML schemes. There are several surveys on collaborative learning *per se* [1], [6]–[10]. Peteiro-Barral *et al.* [6] summarize the approaches for combining predictions of a set of classifiers, including decision rules, stacked generalization, and so on. Devi [7] summarizes the methods related to collaborative classifier learning, collaborative association rule mining, and collaborative clustering. A recent work by Gu *et al.* [8] presented a comparison of distributed ML on mobile devices. Ben-Nun *et al.* [9] focused on distributed DNN, and Verbraeken *et al.* [10] explored communication structures and optimization for distributed learning. Compared with the above surveys, our work focuses on the robustness and privacy issues in collaborative ML. Federated learning has attracted several recent surveys [11]–[14]. Li *et al.* [14] describe vertical and horizontal federated learn-

Manuscript received 12 January 2021; revised 14 January 2022; accepted 1 April 2022. Date of publication 26 May 2022; date of current version 1 December 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62072485 and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515011294. *(Corresponding author: Junbo Wang.)*

Junbo Wang is with the Guangdong Provincial Key Laboratory of Intelligent Transportation System, School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 510275, China (e-mail: wangjb33@mail.sysu.edu.cn).

Amitangshu Pal is with the Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India (e-mail: amitangshu@cse.iitk.ac.in).

Qinglin Yang and Kaiming Zhu are with the School of Intelligent Systems Engineering, Sun Yat-sen University, China (e-mail: yangqlin6@mail.sysu.edu.cn; zhukm3@mail2.sysu.edu.com).

Krishna Kant is with the Department of Computer and Information Science, Temple University, Philadelphia, PA 19122 USA (e-mail: kkant@temple.edu).

Song Guo is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: song.guo@polyu.edu.hk).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2022.3169347.

Digital Object Identifier 10.1109/TNNLS.2022.3169347

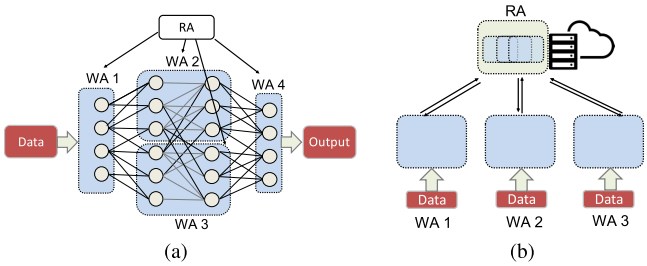


Fig. 1. Two types of parallelism for collaborative ML. (a) Model parallelism. (b) Data parallelism.

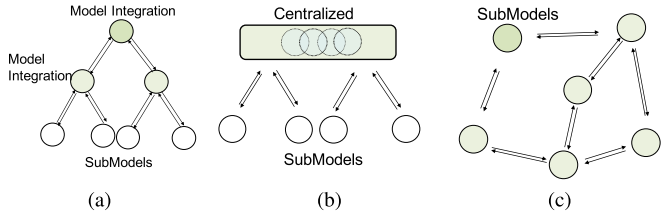


Fig. 2. Three types of model integration: (a) incremental, (b) centralized, and (c) fully distributed.

ings. Peter *et al.* [11] address many aspects of federated learning including efficiency, effectiveness, privacy, and fairness, and highlight open problems for each part of the research. They present a brief introduction to numerous techniques, but the coverage of each is quite limited, especially for privacy and robustness issues. Wang *et al.* [12] focus on the optimization problem in federated learning considering system constraints. The other works, such as [13] and [14], focus on the architecture and algorithms.

The structure of the rest of this article is given as follows. Section III gives a brief introduction to collaborative ML. Section III summarizes different types of collaborative ML. Sections IV and V then summarize the research on robustness and privacy-related issues, respectively. Finally, concludes this article and discusses future challenges. Nomenclature includes the key abbreviations used in this article. We have also included supplementary material on the details of collaborative ML and its applications that the reader may wish to consult. However, we believe that the overview of basic concepts in Section III should suffice to read this article.

II. Collaborative Machine Learning

A. Basic Ideas

A generic component in all these frameworks is that of an RA seeking the help of a number of WAs. A WA may work with an SM or the entire model. The latter case corresponds to the federated learning situation, where the purpose of the distribution is to train the model on different datasets collaboratively without any WA having to reveal its dataset. More generally, the RA may partition the model into multiple SMs to be trained or run by WAs by exploiting model parallelism or data parallelism, as shown in Fig. 1. The RA integrates the results of these SMs using a higher level model, which could range from a simple aggregation of SM results to being inputs to another ML model run by RA. For instance, different SMs may focus on different features or learning aspects, which are then integrated by the RL. A concrete example of this is where each SM recognizes a separate language in a multilingual document. We assume that the integrative model is only known to the RA and itself free from any vulnerabilities. However, we do allow the model training to be iterative in that the RA may provide feedback to the WA's based on the last iteration so that they improve the model or the training. For example, in federated learning, the RA updates the model and provides it to all WAs for further training.

The research on collaborative ML started with distributed/parallel learning, which was investigated to simply make the learning faster by using either *model parallelism* or *data parallelism*, as illustrated in Fig. 1. Model parallelism-based learning separates the whole learning model into several subparts and performs model training separately with the same dataset, while data parallelism-based learning performs model training with different distributed datasets and, finally, aggregates multiple local models in a root node. Distributed ML covers an almost full range of unsupervised learning and supervised learning.

There are several types of collaborative ML frameworks illustrated in Fig. 2. The first one is Fig. 2(a) incremental model integration (IMI), where submodels (SMs) are integrated hierarchically. Take clustering as an example, local data in the leaf node are aggregated first, and the aggregated data are sent to its parent node located in different places for further data aggregation and, finally, achieve a global model in the RA. An example can be found in our previous research [15]. The second is centralized model integration (CMI), as shown in Fig. 2(b), which represents the one-step integration of all SMs as in federated learning. In general, the collaborative training framework can be represented as fully distributed model integration (FDMI), where the integration follows some general acyclic graph, meaning that certain SMs may be integrated along multiple paths. The underlying assumption is that each node is an independent party and has its own computing and storage resources to help with the training and/or inferencing.

The main differences between parallel learning, distributed learning, and federated learning can be summarized as follows.

1) In parallel learning, the whole dataset is split into several small subdatasets, and the SMs trained with subdatasets are mostly aggregated together mainly following master–slave model [see Fig. 2(b)].
2) Distributed leaning targets the scenario where the data are originally collected from distributed nodes, and the aggregation model can be master–slave or fully distributed as a mesh network [see Fig. 2(a)–(c)].
3) Federated learning does not assume that the data distribution is i.i.d. (unlike the other two models). It is also motivated by security and privacy rather than parallelism.

III. Types of Collaborative ML Models

A. Collaborative Linear Regression Models

Linear regression (LR) and the related logistic regression are fundamental methods with a linear estimator. Suppose that we have a training dataset with d features $\boldsymbol{x} = (x_1; x_2; \ldots; x_d)$, and x_d is the value of the vector $\boldsymbol{x}$ in the d dimension. A linear model tries to learn the following linear hypothesis

from the data:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b \tag{1}$$

while minimizing the square loss to find suitable setting of $(\boldsymbol{w}^*, b^*)$ as follows:

$$\left(\boldsymbol{w}^*, b^*\right) = \min \sum_{i=1}^{m} (f(\boldsymbol{x}_i) - y_i)^2 \tag{2}$$

where m represents the number of samples and y_i is the label for item i. Since this function is convex, the closed-form solution can be computed as

$$\boldsymbol{w}^* = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}. \tag{3}$$

Current multiparty collaborative LR [16], [17] mainly collects $(\boldsymbol{X}^T \boldsymbol{X})^{-1}$ from the collaborating nodes and then aggregates them in a root node. The aggregation can also be done in a distributed manner when adopting gradient descent algorithm in ML [18].

B. Collaborative Support Vector Machine

In this section, first, we briefly review the foundation of SVM and then discuss two types of collaborative schemes, each with a different learning scheme.

1) Foundation of Support Vector Machine: The SVM performs a binary classification by finding a hyperplane to maximize the separation of a set of samples into two groups. Given a set of training sample data $\mathcal{X} = \{\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i \in \mathbb{R}^d\}$, where $\boldsymbol{x}_i$ is a feature vector for training the learning model and y_i is the label, the problem can be formalized as follows:

$$\begin{aligned} \min_{\boldsymbol{w},b} \ & f(\boldsymbol{w}, b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \varepsilon_i \\ \text{s.t. } & y_i\left(\boldsymbol{w}^T \boldsymbol{x}_i + b\right) \geq 1 - \varepsilon_i \\ & \varepsilon_i \geq 0 \end{aligned} \tag{4}$$

where b is the bias, $\boldsymbol{w}$ is a weight vector, and C is a regularization hyperparameter that determines the tradeoff between margin maximization and regularization, i.e., training error minimization. The above problem is intractable, and generally, it is reformed as a quadratic programming (QP) problem as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T 1 \\ \text{s.t. } & 0 \leq \alpha_i \leq C \\ & \boldsymbol{y}^T \boldsymbol{\alpha} = 0 \end{aligned} \tag{5}$$

where $Q_{ij} = y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j$, and $\boldsymbol{\alpha}$ is a vector of the Lagrangian multiplier. The weight vector $\boldsymbol{w}$ has relation with $\boldsymbol{\alpha}$ that $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i$. Then, the problem can be solved by solvers for QP problem or sequential minimal optimization (SMO), which is more efficient with small size of $\boldsymbol{\alpha}$ for each iteration.

There are two kinds of designs to extend centralized SVM to a distributed manner: Type 1: parallel design of centralized SVM and Type 2: collaborative SVM for distributed data. Type 1 solutions decompose the whole dataset into several subsets and then process data in a parallel way to speed up the learning procedure. Typical works include cascade SVM [19], divide-and-conquer SVM (DC-SVM) [20], and parallelizing SVM (PSVM) on distributed computers [21].

Type 2 schemes consider a scenario where there are a lot of computation nodes distributed in a specific area. They have abilities to both collect data and train on local data to generate immediate data (e.g., local support vectors). Several approaches are relevant in this case. Flouri *et al.* [22] implement collaborative SVM to fit a distributed sensor network system. They propose a distributed fixed-partition algorithm (DFP-SVM) where the separating hyperplane is obtained through a sequence of incremental steps where each incremental step takes place in a given cluster of sensor nodes. More specifically, in each cluster of sensor nodes, the sensing data are processed as the corresponding estimated hyperplane (i.e., support vectors and offset), and then, the data are transferred to the next cluster. In the end node, the estimated hyperplane is aggregated incrementally. Thus, the sensing data in the previous clusters can be compressed so that the transmission data size between the sensor nodes can be reduced.

C. Collaborative Decision Tree

DT is another popular supervised learning method, which consists of nodes, branches, and leaves. The DT is learned from the training data gradually and can classify a test data layer by layer from the root node to a leaf node. The leaf node decides the classification/regression result of the testing data, and the route from the root node to the leaf node represents the judgment process-based features of the data. In a distributed scenario, PLANET [23] uses MapReduce to distribute and scale tree induction to a very large dataset. For a simplified description, each distributed node computes sufficient statistics based on the local data, and then, a root node aggregates them from all the distributed workers in the reduce procedure. Most works [23]–[25] adopt horizontal partitioning, i.e., they partition the data. In contrast, YGGDRASIL] [26] adopts vertical partitioning based on the features, i.e., each worker stores feature values for an even number of features, as well as the labels for the instances for training.

With the development of federated learning, DT also is redesigned for the federated platform [27]–[29]. Liu *et al.* [27] propose federated forest algorithms working for client and server, and the tree structure is stored on the master node and every client. For each round of federated learning, each client selects the best features based on the local data, whereas the master node selects all the responses from the clients and does aggregation. Then, the master node selects the best features and notifies all the clients. Liu *et al.* [28] propose FEDXGB for extreme gradient boosting (XGBoost) working in the federated learning framework. FEDXGB follows the basic scheme of federated learning, consisting of a set of workers and a root node. The workers send gradients to the root node with *homomorphic encryption*, and the root node interactively generates classification and regression tree (CART) by finding the optimal split feature. Li *et al.* [29] also implement gradient boosting DTs (GBDTs) in federated learning framework.

D. Federated Learning

The collaborative framework provided by federated learning is suitable for many types of learning algorithms, such as linear models, DT, and so on, although NN is most typical applications. The framework involves running the same model on multiple worker nodes and averaging the weights from them as follows:

$$\boldsymbol{w}^{(i+1)} = \boldsymbol{w}^{(i)} - \eta\left(\frac{1}{n}\sum_{j=1}^{n} g\left(\boldsymbol{w}_j^{(i)}\right)\right) \tag{6}$$

where $\boldsymbol{w}^{(i)}$ is the parameters used in the ith round, n is the number of workers in the ith round, and $\boldsymbol{w}_j^{(i)}$ represents the parameter updated by the jth worker in the ith round. After aggregation, the server distributes $\boldsymbol{w}^{(i+1)}$ to all workers for the next round of training. With the continuous training and aggregation process, the procedure will stop when certain global accuracy is achieved. To further reduce the communication burden between a client and the aggregation server, *structured updates* and *sketched updates* are proposed in [36] to reduce the number of variables and compress them before sending them out. *Structured update* modifies gradients in a restricted space such as using a low-rank update matrix or restrict the update by a random sparse matrix. *Sketched update* first computes the full gradient matrix during local training and then compresses it before sending it to the root node. Federated stochastic variance reduced gradient (FSVRG) was proposed in [37] to be adaptive to different local data sizes and different patterns of generated local data. The basic idea of SVRG is to update the weights based on the variance of gradients during the iteration, while FSVRG implements SVRG in the federated learning scheme.

Recent works have shown that federated learning can converge after several iterations [50], [51] although the iterations among local and global learning nodes consume much more time [52]. Federated learning can be used in many different scenarios, such as credit rating and smart medical health [14]. WeBank and TensorFlow have proposed their own development framework called FATE and TensorFlow federated, and many researchers are interested in its application in more scenarios, such as resource-restrained IOT devices [52]–[55] and non-i.i.d. data [50], [56]–[58]. Although federated learning is designed for privacy-preserving, information leakage is still possible by gradient exposure during the learning procedure, which we address in Section V.

E. Collaborative Clustering

The most popular unsupervised learning is clustering. Collaborative clustering works as follows: each worker node generates a local cluster from the available data possibly based on different policies. The local clusters are then aggregated in a root node.

There are several policies to represent local clusters by using: 1) the core points inside of a cluster; 2) the specific core points; and 3) the boundary points in the clusters. The most basic method is to represent a cluster by the points inside of each cluster [44], [45], [59], [60], which can be seen as core points in a region. For example, in a density-based scan (DBSCAN), core points are selected to represent a cluster if it is in a given radius (*EPs*), at least a minimum number of points (MinPts) in the cluster.

DBSCAN has been extended to work in a distributed/parallel way. In density-based distributed clustering (DBDC) [42], the global cluster is reconstructed based on the aggregated information from the worker nodes. PartDBSCAN [44] uses a master–slave model and adopts a dR-tree to balance the partitioning of data among the slave nodes that do the clustering and return them to the root node for the merger. Such an approach can also be applied to other types of clustering, e.g., grid-based clustering in [15].

Specific core points are the points selected from the core points to further reduce data transmission in collaborative clustering. In DBDC [42], Scor_C is proposed as the complete set of specific core points satisfying the following two conditions: 1) any pair of points in Scor_C is not located within the EPs-neighborhood of each other and 2) for each core point c, there is at least one specific-core point s that c is within EPs-neighborhood of s. The set of points that represent the minimal set of core points from a single cluster is used for data reduction in [61] and [62]. In their studies, eight points can represent the core points of a grid cell for an arbitrary density. Specific core points reduce the data size; however, the accuracy of clustering results can be decreased if the selections are not suitable.

Another approach to represent local clusters is to use boundary points as representative points, as studied in [46] and [47]. The local dataset is clustered, and contours are found for each local dataset once the local clusters are determined. Then, worker nodes exchange their contours with their neighboring nodes and see whether there are overlapping contours. Finally, global clusters are created based on the merging of overlapping contours. The selection policy based on boundary points can be seen as a compromise solution between core points and specific core points, and the data size is directly proportional to the size of the cluster.

Table I compares various models in terms of collaborative model classification shown in Fig. 2, information transmission, and other attributes. Only one of these uses model (c); all others use (a) or (b). The information transmission between RA and WAs includes model parameters or raw data. The model parameters depend on the ML algorithms, e.g., Matrix X for the linear model, subvectors for SVM, splitting information for DT, gradients/weights for NN, and core points for the cluster.

F. Other Types of Collaborative ML

1) Event-Triggered Learning: Event triggering refers to information or messages that can trigger actions or responses of agents in a distributed system. Events may be related to time or steps and are used to control communications between distributed agents. Several event-triggered mechanisms have been used to improve the convergence by controlling the update frequency. For instance, George *et al.* [63] use it to solve nonconvex optimization problems typically encountered in distributed deep learning, such as image classification. They propose a Distributed Event-Triggered Stochastic GRAdient Descent (DETSGRAD) algorithm where the WAs update the

TABLE I
RESEARCH ON COLLABORATIVE LEARNING

Types of ML	Collaborative Learning		Model	Information Transmission	Robustness, Privacy, Learning Quality, Complexity or other Remarks
Linear Model	Privacy-preserving Linear Regression [17]		(b)	Matrix X	Secure and better in larger dimension; ACC: $0\%-5.2\%$ different with optimal
	Secure Logistic Regression [18]		(b)	Encrypted raw data	Homomorphic encryption; ACC: $0\%-2.4\%$ different with optimal
SVM	Cascade SVM [19]		(a)	subvectors	Binary tree-based structure, ACC:99%
	DC-SVM [20]		(b)	subvectors	Divide and Conquer, ACC:96%, Complexity: $\mathcal{O}(n^2/k)$
	PSVM [21]		(b)	Parallel ICF	Linear complexity with n, ACC:95.9%, Complexity: $\mathcal{O}(kn)$
	QRSVM [30] [31]		(b)	Matrix R after QR decomposition	Linear complexity with n, Complexity $\mathcal{O}(kn)$
	Fully distributed SVM [32]		(c)	Extreme points of local convex hulls	Fully distributed structure
Decision Tree	**Distributed**	Privacy-DDT [33]	(b)	Encrypted Gini Index	Homomorphic encryption, ACC close to optimal
		DDTv2 [24]	(b)	Local trees	Hadoop and Spark, ACC:97.16%
		PLANET [23]	(b)	Separated data	Map-reduce framework
		YGGDRASIL [26]	(b)	Split information	Faster than [23] by an order of magnitude
		DR2-Tree [25]	(b)	Raw data	Spark, ACC: $0\%-0.27\%$ different with optimal
		InPrivate Digging [34]	(a)	Tree model	Differential privacy, ACC: $0.08\%-0.1\%$ different with optimal
	Federated	FEDXGB [28]	(b)	Gradients	Homomorphic Encryption, 1% ACC loss, 33% communication reduce
		Federated Forest [27]	(b)	Split message	homomorphic encryption, ACC $0.01\%-0.08\%$ different with optimal
		SimFL [29]	(a)	Tree model	Locality-Sensitive Hashing, ACC $0\%-2.1\%$ different with optimal
Federated Learning	FedAvg [35]		(b)	Model parameters	Federated averaging, ACC:99.44% after 300 rounds
	F-Updates [36]			Model parameters	Reduce communication cost by two orders of magnitude by structure/sketched updates
	F-SVRG [37]			Model parameters	Assume the setting of large size of client, converge in 30 rounds with error 0.26%
	FedMeta [38]			Meta-learner	Increase converge time by 2.82-4.33x and ACC around $3.23\%-14.84\%$ comparing to FedAvg
	FedMA [39]			Model parameters	Matching and averaging hidden elements for NN, ACC 0.87% in 100 rounds on VGG-9.
	Agnostic Federated Learning (AFL) [40]			Model parameters	The model will not be biased towards different clients, ACC increase 0.32% to traditional FL
	Agnostic meta-learning [41]			Model parameters	Personalized FEDAVG
Clustering	**Density**	DBDC [42] [43]	(b)	Core points	Represent cluster by core points, ACC: almost the same to centralized cluster, Complexity: $\mathcal{O}(n^2)$
		PartDBSCAN [44]	(b)	Core points	R-Tree based structure, ACC: almost the same to centralized
					cluster, Complexity: $\mathcal{O}(\lvert S\rvert \log n)$
		MrDBSCAN [45]	(b)	Core points	Map-Reduce based scheme, Complexity: $\mathcal{O}(\lvert S\rvert \log n)$
		Grid-Based [15]	(a)	Grids	Hybrid approach, ACC: close to centralized cluster
		Boundary based [46] [47]	(b)	Boundary points and density	Aggregation based on boundary points, ACC: close to centralized cluster
	Partition	PK-Means [48]	(b)	Centroid list	Message-passing based cluster, ACC: close to centralized cluster
		PPK-Means [49]	(b)	Parameters	Privacy model, ACC: close to centralized cluster

model parameters aperiodically with their one-hop neighbors. The mechanism also provides sufficient conditions on the algorithm step sizes that guarantee the asymptotic mean-square convergence with complexity $\mathcal{O}(1/((k+1)^{\delta_2}))$, where k denotes the time instant and δ_2 denotes the hyperparameter. Fan *et al.* [64] propose an iterative event-triggered algorithm to avoid continuous measurement of the neighbor's states, which can decrease the communication frequency for the distributed rendezvous issue of multiagent systems. The main idea of this approach is to determine the measurement error

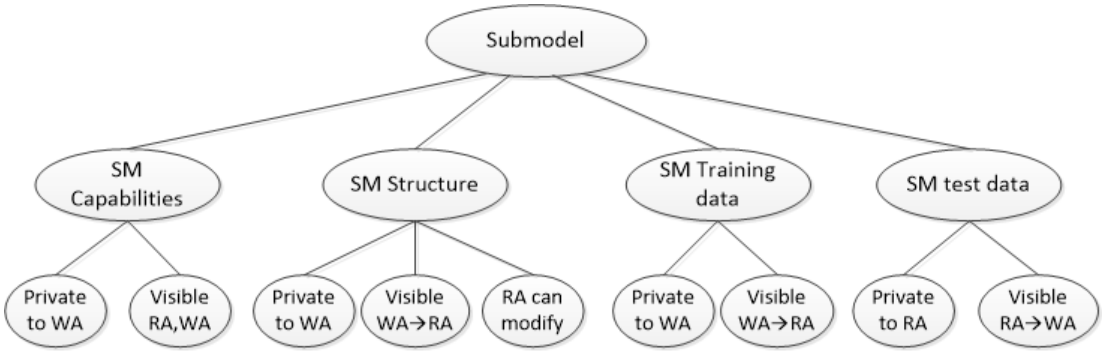


Fig. 3. Illustration of SM parameters.

of each agent by a convex combination of its neighbors' states instead of measuring the agents' own states, because Fan *et al.* [64] find that each agent will be triggered regularly, and the group will asymptotically achieve rendezvous by classifying the triggering executions.

2) Semisupervised Learning (SSL): It emphasizes how to achieve learning with the presence of both labeled and unlabeled data. SSL exploits the fusion of labeled and unlabeled data to change the learning paradigms and designs related algorithms that can take advantage of such a fusion [65].

Several works have been done to enable SSL work in a distributed/parallel way. For example, Chang *et al.* [66] study the error analysis issue for distributed SSL algorithm, kernel ridge regression (DSKRR), by using a novel error decomposition scheme that partitions the generalization error of DSKRR into three parts: approximation error, sample error, and distributed error. Bilmes *et al.* [67] discuss the scalability to very large problems sizes (>100 million nodes) of graph-based SSL algorithms on diverse parallel machines, either shared-memory symmetric multiprocessors (SMPs) or distributed computers.

3) Multitask and Multiagent Learning: Here, multiple tasks share the trained model and are divided into parameter-based sharing and constraint-based sharing, which allows multiple related tasks to be trained simultaneously. For instance, Smith *et al.* [68] demonstrate that multitask learning [69] is naturally suitable for the statistical challenges (non-i.i.d. issues). They propose a novel systems-aware optimization method for practical systems issues (client heterogeneity), named "MOCHA," which considers issues of high communication cost, stragglers, and fault tolerance. MOCHA is a biconvex alternating approach that can be guaranteed [70] to converge to a stationary solution to the MTL problems.

In a multiagent system [71], agents solve tasks collaboratively yet offer more flexibility since they can learn and make autonomous decisions. The agents learn new contexts and actions by leveraging their interactions with neighboring agents or with the environment and address their allocated task by using their knowledge to decide and perform an action on the environment.

IV. Robustness of Collaborative ML

Robustness of collaborative ML has been discussed widely in recent literature, mostly in the context of federated learning [2]–[5]. To make the discussion in a more concentrated way, we mainly focus on the robustness from a security aspect. In general, an ML SM has four important attributes that may or may not be shared (in addition to the basic API for using the SM, which we assume is always shared). These are shown in Fig. 3 and explained in the following.

1) *Submodel Capabilities:* It specifies what nuances of the real world (e.g., variations, imperfections, or irregularities in the objects, features, or ambient conditions) that the model can discriminate. These could be either private to the WA (possibly because WA owns the model) or shared between WA and RA. In the latter case, these could be either provided by the RA to WA as requirements for a private model built by WA or voluntarily shared by WA with RA.
2) *Submodel Structure:* It specifies low-level details of the SM. For example, in the case of a DT, the structure consists of the entire tree, and in the case of a CNN, all the layers and weights are part of the structure. The structure could either be private to the WA, made visible to RA by WA, or modifiable by RA. In the second case, WA simply builds, trains the model, and then submits it to RA for validation. In the third case, the RA can modify the received SM further and sends it back to WAs for further training. The finalized model may be retained by WAs for the purposes of running it on WA's computing infrastructure.
3) *Submodel Training Data:* The ability of a WA to train the model for RA without revealing the training data forms the fundamental motivation for the federated ML. However, since the training of a DNN is often an extremely compute and data-intensive task, training by WA can be valuable even if there is no prohibition on sharing the training data.
4) *Submodel Test Data:* In all cases of distributed ML, we assume that the RA will not accept an SM (or results thereof) unless it can pass a specified set of tests on a challenge dataset provided by the RA. If the SM structure is shared by WA (i.e., a trained SM delivered by WA to RA), the test data will likely be kept private by the RA. However, it is also possible that the RA provides this data to WA and expects the test results

back. (According to our assumption, the WA provides the test results truthfully in this case.)

The new robustness issue in this model is that a few of the WAs may be malicious or untrustworthy, or their communications with the RA may be compromised by man-in-the-middle or other types of communication attacks. While the communication vulnerabilities can be addressed using standard techniques (e.g., message encryption and authentication of WAs), the insider attacks by malicious/compromised WAs can be much more difficult to tackle.

A. Submodel Information Exchange Modes

The collaborative model admits different scenarios in terms of information exchange between the RA and WAs, and the corresponding vulnerabilities. The vulnerability depends on several aspects, including: 1) the extent of data/model sharing among RA and WAs and the attacks enabled by this sharing; 2) the attacker capabilities (i.e., level of maliciousness assumed for WAs); 3) the possibility of collusion among WAs; and 4) validation or compromise detection capabilities of the RA. This leads to a large number of potential scenarios, each of which is further distinguished by the nature and parameters of the ML models used by the WAs. Due to the large state space, in the following, we only sketch a possible taxonomy of situations and the types of attacks that it may admit. A part of the taxonomy is shown in Fig. 3. Here, we assume that, if WA and RA explicitly stated to share some information or specification, sharing is considered a "contract" and remains uncompromised.

Fig. 3 points to many specific scenarios and the corresponding vulnerabilities. Consider, for example, the case where all four items are shared between RA and WA. This corresponds to the situation where the RA provides the requirements to the WA, according to which the WA builds, trains, tests, and then delivers the final SM to the RA that satisfies all the validation tests. In this case, one potential vulnerability pertains to the specification of capabilities by RA. A bad WA can take advantage of this knowledge and train the SM on some additional data that only affect the aspects of the model not covered by the "contract" (or specification of capabilities by RA). The knowledge of validation tests also helps in that it can ensure that all tests still pass. Note that less sharing by RA makes the job difficult for both good and bad WAs. For example, if the RA does not share the SM capabilities or the test data with WA, a bad WA does not know how to train the model on bad data without the compromise being revealed. By the same token, a good WA will also have difficulty in meeting the expectations of the RA. In contrast, less sharing by WA (with RA) provides greater opportunities for bad WA to perturb the model or the results. For example, if a WA does not share its training data with RA, a bad WA is free to train the model on anything so long as the model provides reasonable results on the validation tests.

Collusion among bad WAs can further amplify the perturbation of the model. In particular, colluding WA's can stagger their spurious output submission in a way to minimize detection. For example, in the case of the same SM trained by different WAs, a set of bad WA's could supply trained weights to the RA in succession in a way that no improvement takes place over successive training iterations. With identical SMs, it might be possible for the RA to detect such collusion using Byzantine fault principles, i.e., when no more than t out of a total of $3t + 1$ WAs are bad.

B. SubModel Information Exchange Versus Attack Modes

Fig. 3 allows for 16 different scenarios of sharing, each of which can be exploited by a bad WA to carry out various forms of perturbation either during training time or run-time. Most of the attack modes are known in the literature under different names, as discussed in the following. Pitropakis *et al.* [72] present a taxonomy of these modes and comprehensively discuss the papers focusing on these modes. These basic modes can be related to the 16 different possibilities in Fig. 3 in terms of the level of visibility (black-, grey-, and white-box attacks).

Given the indirect relationship between training data and the model parameters (e.g., weights in CNN) and the need for a large amount of data for training, the model can be perturbed by strategically providing bad data at training time. These could take several forms. A *data poisoning attack* changes the labels on some of the valid data items to force misclassification in certain cases. An *evasion attack* takes advantage of the fact that the model may have weak points that can be exploited, i.e., the inability to correctly recognize certain features, which are then added deliberately to the inputs at inference time. If the attacker can, it may enhance or create this weakness deliberately by withholding certain training data. Note that a simple form of evasion might only increase the classification uncertainty, which can be achieved by withholding some training data or adding what amounts to noise to the data. *Backdoor attacks* intelligently add extraneous data that make the model generate anomalous output under certain scenarios. These scenarios are then triggered at inference time by tampering with the input. The backdoor attack is particularly insidious when the data obtained during inference time come directly from items in the physical world that can be easily tampered with. For example, if a special physical pattern (e.g., a bar code) on the rear of a car is the backdoor, all that the adversary has to do is to paste that pattern on the rear of some cars (including the car(s) belonging to the adversary). This could make the car just behind it misbehave (e.g., driving too close or crashing into it). The same applies to manipulating physical traffic signs, such as forcing a low-speed limit sign to something that indicates a much higher speed limit.

Model poisoning attack concerns changing the output of the model (either using poisoned data or in some other way) to force misclassification. Model poisoning happens if the training involves iteration; i.e., the RA collects models or model outputs from multiple WA's and uses them to provide feedback to the WAs. For example, if the feedback that depends on the output of one bad WA goes to all the WAs for the next iteration, the entire model gets corrupted or poisoned. Bhagoji *et al.* [2] discuss several such attack strategies, including targeted model poisoning using boosting of the malicious agent's update to overcome the effects of other

agents. They also show that Byzantine-resilient aggregation strategies are not robust to these attacks.

Yet, another type of attack is *model stealing attack*, where a WA is able to deduce the SM belonging to another WA even though there is no direct revelation of other WA's SMs. The source of this attack is the feedback by RA that includes feedback based on other SMs.

With respect to Fig. 3, it is obvious that various attacks are enabled by two aspects of the system: 1) sharing of information by RA with bad WAs and 2) ability of a bad WA to do things beyond the contract (e.g., training with extraneous data).

C. Data Poisoning Attacks

Poisoning attacks were first proposed for faking the biometric recognition system. Biggio *et al.* [73], [74] have discussed how such attacks can gradually poison the template gallery to successfully mislead a template self-update-based face-verification system. In template self-update-based face-verification system, a template corresponding to each client is stored by averaging a set of n enrolled images, which is referred to as *centroid*. If the feature vectors corresponding to client c are $\{x_{c1}, x_{c2}, \ldots, x_{cn}\}$, then their centroid is $x_c = ((\sum_{k=1}^{n} x_{ck})/n)$. When a user submits a sample x, a matching score $s(x, x_c)$ is computed as

$$s(x, x_c) = 1/(1 + ||x - x_c||) \tag{7}$$

where $||\cdot||$ is denotes the Euclidean distance. If the matching score is greater than a certain threshold t_c, the sample is accepted as genuine; otherwise, it is rejected. To update the centroid over time, the authors have discussed two policies. The first is the *infinite window* policy, where the centroid x_c is updated without discarding any of the past samples, and the second policy is *finite window* where the samples in the last n iterations are only considered for centroid calculation.

Template self-update is implemented to deal with temporal changes in biometric patterns, such as aging. The self-update is accepted if $s(x, x_c) > \theta_c$, where w_c is the update threshold and is generally greater than the matching threshold t_c. By exploiting this self-update feature, the poisoning attack injects specifically targeted samples that are accepted by the system as normal and push the centroid x_c in the direction of the attack point x_a. Biggio *et al.* [73] have studied that, in the infinite window scenario, the number of attack samples must grow exponentially with $||x_a - x_c||$, whereas, in the case of the finite window, the number of samples must grow linearly. Thus, the latter is more vulnerable to poisoning attacks compared to the former scenario. Poisoning attacks are also studied for anomaly detection in [75] and [76].

Szegedy *et al.* [77] have shown that, by applying an imperceptible nonrandom perturbation to a test image (which are termed *adversarial examples*), it is possible to fool a DNN to arbitrarily change the network's prediction. Goodfellow *et al.* [78] have argued that the primary cause for such vulnerability to adversarial perturbation is their linear nature. Similar poisoning attacks on SVMs are studied in [79] and [80]. Chen *et al.* [81] have shown that, by injecting around 50 dirty-label samples, a backdoor adversary can achieve an attack success rate of above 90%.

Yang *et al.* [82] have discussed data poisoning attacks on NNs using direct gradient methods. They have also proposed a generative method to speed up the generation of poisoning data by using the inspiration from GAN and have shown that the generative method speeds up the poisoned data generation by 239.38 times compared to the direct gradient method. Muñoz-González *et al.* [83] have proposed a *back-gradient optimization* to launch poisoning attacks on NNs and deep learning architectures, which is more computationally efficient than gradient-based poisoning attacks. Zhang *et al.* [3] have studied poisoning attacks on federated learning systems based on GAN, where an attacker can stealthily train a GAN to mimic the prototypical samples of the other workers. The GAN can then be controlled for generating the poisoning updates.

D. Model Poisoning Attacks

Several types of model poisoning attacks are possible in collaborative ML. In the *outsourced training attack*, the RA wants to train the parameters of a DNN F_Θ (where Θ represents the function's parameters), using a training dataset D_{train}. The WA trains the model and returns the trained parameters Θ'. The RA checks the accuracy of the trained model $F_{\Theta'}$ on a (labeled) validation dataset D_{valid} and will only accept the model if the accuracy of the model is more than a certain threshold a^*, i.e.,

$$\mathcal{A}(F_{\Theta'}, D_{\text{valid}}) \geq a^*. \tag{8}$$

In this case, the WA can return a maliciously backdoored model such that: 1) the returned model does not reduce the classification accuracy of the validation set and 2) for certain inputs containing the backdoor trigger, the prediction of the outputs is different from the prediction of the honestly trained model. The backdoor trigger could be either explicit thereby requiring perturbation of real data to exploit the backdoor (e.g., by infecting some queries), or it could be implicit: the model produces bad output on certain types of data that the attacker believes would not be a part of test data. Even if the model fails verification, but the RA is willing to let the WA "fix" the problem, it provides a mechanism for the malicious WA to check what kind of backdoor to not use.

Shen *et al.* [4] have studied similar poisoning attacks in collaborative learning settings where different users submit the masked features to a central classifier to learn the global model. They have shown that, in such a setting, just by poisoning 10% of the training data, the attacker can achieve misclassification with a success rate of 99%. Bagdasaryan *et al.* [84] have developed a model poisoning attack where the malicious worker can use model replacement to introduce backdoor functionality. The authors have shown that such a model replacement attack greatly outperforms the conventional data poisoning attack.

Cao *et al.* [5] show that the success rate of model poisoning increases linearly with the number of poisoned samples and the number of attackers. This article also proposes a filtering defense based on the distance between model updates by honest and malicious agents. Hayes *et al.* [85] discuss a variant

of model poisoning attack, called data contamination attack, which uses data specifically to learn undesired correlations. For example, a model to determine if a credit card application should be accepted could maliciously discriminate against applicants of certain characteristics. This article shows that adversarial training can mitigate such attacks. Another variant is a form of the Sybil attack discussed in [86] where the malicious agent mimics the actions of a legitimate agent but alters the training data to force misclassification.

E. Black-, Grey-, and White-Box Attacks

Adversarial ML is often broadly categorized into black-, gray-, and white-box attacks [87], based on whether the attacker, respectively, has zero, partial, and full knowledge about the ground truth and the learning mechanism. The zero-knowledge situation is known as *Blackbox*, the knowledge of both the learning mechanism and the ground truth is known as *Whitebox*, and the knowledge of one of these is known as *Greybox*. A Greybox characterization is not very useful unless the extent of knowledge is quantified, as is done in our model in Fig. 3. Note that anything that a WA can keep private could potentially be altered by it, possibly maliciously. Even in the case of visibility, it is possible that what is shared is not what is used, but we generally assume that there are no contractual violations (e.g., if RA and WA agree to use certain data for training, it is indeed used) since, otherwise, a classification is not very meaningful.

Carlini *et al.* [88] have studied black- and white-box threat models on MNIST and CIFAR-10 datasets in the context of image classification. They have studied the existing detection techniques for these two threat models and have shown how to choose a good attacker loss function for each defense. Chivukula *et al.* [89] have discussed the interaction between an adversary and a deep learning model as a two-player sequential noncooperative Stackelberg game with the assumption that the adversary does not know anything about the network structure. In this game, the learner learns the weights of a CNN for correct classification, and the adversary creates new instances of data using genetic operations for misleading the classification. The game is solved by the Nash equilibrium, which leads to solutions that are robust to subsequent adversarial data manipulations. Papernot *et al.* [90] have introduced black-box attacks against DNN classifiers where the only capacity of the attacker is to observe labels assigned by the DNN for the chosen inputs and has shown that the DNN misclassifies 84.24% of the adversarial inputs. Papernot *et al.* [91] have developed generalized black-box attacks by exploiting adversarial sample transferability on broad classes of ML classifiers, including NNs, logistic regression, SVMs, DTs, nearest neighbors, and ensembles. Dong *et al.* [92] have proposed a broad class of *momentum iterative gradient-based methods* to boost the success rates of the generated adversarial examples and have shown that their iterative methods exhibit higher success rates in both white- and black-box attacks.

Eykholt *et al.* [93] have discussed a general attack algorithm named *Robust Physical Perturbations (RP_2)* to generate adversarial perturbations under white-box settings and have shown that RP_2 achieves high target misclassification. The authors have shown that their attacks cause a standard road sign classifier to interpret a slightly modified stop sign as a Speed Limit 45 sign.

Papernot *et al.* [94] have used the concept of "defensive distillation" as a defensive mechanism against adversarial samples by reducing the amplitude of NN gradients that are generally exploited by the adversaries to craft adversarial samples. The authors have empirically studied that such defensive distillation reduces the attack success rate from 95% to less than 0.5% on the MNIST dataset. However, Carlini *et al.* [95] have created a set of white-box attacks that successfully find adversarial examples for 100% of images on defensively distilled networks. Papernot *et al.* [96] have discussed a class of algorithms for adversarial sample creation against any feedforward DNN with the assumption that the adversary has knowledge of the network architecture and its parameter values. They have shown that, by modifying only 4.02% of the input features per sample, their algorithms can misclassify specific targets with a 97% adversarial success rate. Other white-box attacks are also studied in [97]–[100]. Adversarial robustness of NNs against both black- and white-box attacks are discussed in [101]. A summary of representative attackers is discussed in Table II.

F. Byzantine Robustness

Federated learning enables SM structure sharing, as shown in Fig. 3, where each worker j trains an SM at local ($w_j^{(i)}$) in the ith round, and the SMs are aggregated at the RA as in (6). During the learning process, the Byzantine workers may send arbitrary or incorrect messages to the RA, which leads to incorrect or nonconvergence of model training. Several Byzantine-robust aggregation methods have been proposed to address this problem, as summarized in the following.

1) Median-Based Aggregation: Median-based aggregation has been investigated in the recent works trying to solve Byzantine attacks [102], [103]. The basic method is given as follows: 1) calculate gradients in each client site; 2) perform gradient aggregation based on different median functions; and 3) update model parameters in the aggregation stage. Take geometric median as an example; let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \subseteq \mathbb{R}^d$ be a set, where $\boldsymbol{x}_m$ is a vector. Geometric median of $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ denoted by $\text{med}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$. It can be calculated as follows:

$$\text{med}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \sum_{i=1}^{n} ||\boldsymbol{x} - \boldsymbol{x}_i||_2. \tag{9}$$

Then, in the second stage, gradient aggregation is performed by jointly considering the above geometric median function, as shown in the following equation:

$$A_k(\boldsymbol{w}_t) = \text{med}\left\{ \frac{1}{|B_1|} \sum_{j=1}^{|B_1|} g_t^{(j)}(\boldsymbol{w}_{t-1}), \ldots, \frac{1}{|B_m|} \sum_{j=1}^{|B_m|} g_t^{(j)}(\boldsymbol{w}_{t-1}) \right\} \tag{10}$$

where $|B_m|$ represents data size in the client m and $g_t^{(j)}(\boldsymbol{w}_{t-1})$ denotes the gradients calculated in the iteration t with sample j. Finally, the aggregation results can be updated through

TABLE II
SUMMARY OF REPRESENTATIVE ATTACKS

Attack Types	Details	Representative Works	Attacker's knowledge	Use cases
Data Poisoning attacks	Attacker changes the labels on some of the valid data items to force misclassification	Reference [73]	Perfect knowledge	Template self-update/face verification
		Reference [74]	Limited knowledge	Face verification
		Reference [75]	Zero knowledge	Classification
		Reference [76]	Full/limited control over training data	Anomaly detection
		Reference [77]	Limited knowledge	Image classification
		Reference [79]	Perfect knowledge	Image classification
		Adversarial label flips attack [80]	Perfect knowledge	Binary classification
		Backdoor attacks [81]	Zero knowledge	Image classification
		Generative Poisoning Attack [82]	Perfect knowledge	Image classification
		Back-gradient optimization [83]	Perfect/limited knowledge	Spam filtering, malware detection, handwritten digit recognition
		Generative Adversarial Nets [3]	Perfect/limited knowledge	Image classification
		Reference [88]	Perfect/zero knowledge	Image classification
Model Poisoning attacks	Attacker changes the model output to force misclassification	Reference [4]	Limited knowledge	Image/traffic sign classification
		Reference [84]	Limited knowledge	Image classification
		Reference [85]	Limited knowledge	Multi-party environment
		Reference [86]	Limited knowledge	Image classification
Black box attacks	Attacker has zero knowledge about the ground truth sand the learning mechanism	Reference [88]	Zero/limited knowledge	Image classification
		Reference [89]	Zero knowledge	Image classification
		Reference [90]	Zero knowledge	Image classification
		Reference [91]	No knowledge	Image classification
		Momentum iterative gradient-based methods [92]	No knowledge	Image classification
White box attacks	Attacker has full knowledge about the learning mechanism and the ground truth	Reference [88]	Perfect knowledge	Image classification
		Robust Physical Perturbations [93]	Perfect knowledge	Road sign/image classification
		Reference [95]	Perfect knowledge	Image classification
		Reference [96]	Perfect knowledge	Image classification
		Reference [98]	Limited/Perfect knowledge	Malware detection
		Reference [99]	Perfect knowledge	Image classification
		Reference [100]	Perfect knowledge	Image classification

the following equation:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta \times A_k(\boldsymbol{w}_t). \tag{11}$$

The work in [102] shows that the proposed geometric-median-based aggregation can tolerate q Byzantine failures such that $2(1+\varepsilon)q \leq m$, $(\varepsilon > 0)$, where m represents the number of working machines.

2) Krum Aggregation: Considering gradients from each client as a vector, the vectors from general clients should basically follow a similar direction, while the vectors from Byzantine clients may be far away from them. The basic idea of Krum [104] is to preclude the vectors that are too far away from normal users. They defined a Krum aggregation function $\text{Krum}(V_1, \ldots, V_n)$, which can return the client i_* by minimizing the score $s(i)$ in the following equation:

$$s(i) = \sum_{i->j} \left|\left|V_i - V_j\right|\right|^2 \tag{12}$$

where $i \to j$ means that V_j belongs to the $n-f-2$ closest vectors to V_i, n represents the number of clients, and f is a random number between 0 and n. Then, network parameters are updated through the following equation:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta \times \text{Krum}(V_1, \ldots, V_n) \tag{13}$$

where at least $n-f$ vectors among n clients are correct and the other ones may be Byzantine, which can be filtered in the learning process.

3) Robust Stochastic Aggregation (RSA): The primary disadvantage of the previous schemes comes from the i.i.d. assumption, which is not the case in federated learning over heterogeneous computing units. In addition, some of these algorithms like Krum rely on sophisticated gradient selection subroutines, which incur a high computational cost. RSA [105] overcomes these limitations. It updates the parameters

(e.g., in an NN) through the following equation:

$$w* = \underset{w=[w_i;w_0]}{\arg\min} \sum_{i\in R} \left(\mathbb{E}\left[F(w_i,\xi_i) + \lambda||w_i - w_0||_p\right]\right) + f_0(w_0) \tag{14}$$

where w_i represents the weights from the client i and w_0 denotes the weights calculated in the master. $F(w_i, \xi_i)$ is the loss function of worker i with respect to a random variable ξ_i assuming that the data across the workers are i.i.d. The $\lambda||w_i - w_0||_p$ term works as the ℓ_p-form penalty to force every x_i to be close to the master's variable x_0.

V. Privacy Issues in Collaborative ML

In addition to the possible attacks on collaborative ML, there are also vulnerabilities with respect to privacy. Recall that the key motivation for federated learning is to maintain the privacy of local data; however, even in this case, the data privacy could be compromised. There are three main types of attack in federated learning, which are membership inference, unintended feature inference, and representative sample reconstruction. In this section, we first introduce these three privacy attacks and discuss current solutions trying to solve them.

A. Membership Inference

The goal of membership inference is to infer if a certain dataset was used in the training. Take a document survey as an example; the adversary who plans a membership inference attack wishes to confirm whether an individual participated in the survey [106].

Shorti *et al.* [107] proposed a framework for membership inference in a centralized learning scenario, and it worked extremely well even in the public ML service, such as Google and Amazon. The basic consideration is that the value in each dimension from the softmax function can represent the probability that the input belongs to that category. The training continuously increases the *model confidence* by reducing the gap between the hypothesized function and the obtained results, which makes the probability of privacy leakage higher and higher. The authors found that the samples in the training set have much higher model confidence than others, which did not join the training procedure, and this can be exploited for membership inference. They use three methods to construct the victim-users' fake dataset, and the membership inference attack shows excellent performance with different datasets and network structures. Yu *et al.* [108] propose an effective and efficient black-box attack methodology that extracts large-scale DNN models from cloud-based platforms with near-perfect performance. Their work dramatically reduces the number of queries required to steal the target model by incorporating several novel algorithms, including active learning, transfer learning, and adversarial attacks, as depicted in Fig. 4. The main idea behind the shadow model is to use input–output pairs obtained by querying the black-box APIs with malicious samples to produce an attack model that could successfully infer membership with smaller querying times. The results show that the alternate model by using

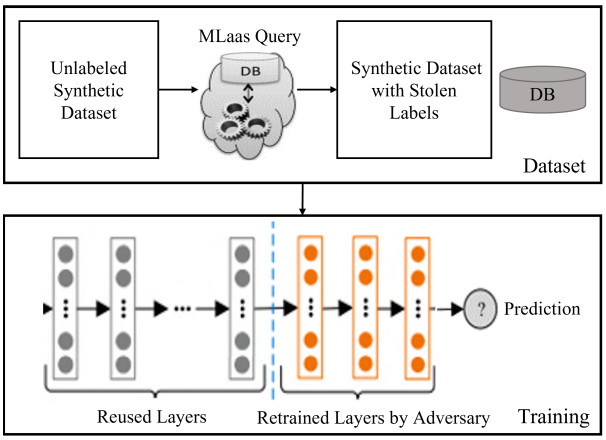


Fig. 4. Illustration of the shadow model.

VGG19 DeepID as the transfer architecture with the proposed algorithm can reach 74.25% accuracy with 2.15k queries than Tramer attacks [109] results (15.97%) under the same conditions. Nasr *et al.* [110] assume that the active attacker can exploit the stochastic gradient descent (SGD) algorithm to run the active attack since the SGD algorithm forcefully decreases the gradient of the training loss and generalizes to the testing loss as well. Their white-box membership inference attack can reach a considerably higher accuracy of 74.3% compared with the 50% of the baseline for the random guess.

However, the reason for the successful attack is still not so clear. Many researchers think that it happens because the NN is *overfit*. In fact, with the *early stop* and *regularization*, the attack in the previous work [107] performs worse but still works better than the baseline of *random guess*. Thus, the extension of this topic can be divided into two approaches: one is to figure out the relationship between overfitting and membership inference attack, and the other is to extend the membership inference in different scenarios. For the first approach, Song *et al.* [111] investigated the overfitting scenario by simulating a malicious ML provider, who provides ML algorithms for other users as services. They evaluated memorizing users' data by an ML algorithm in both white- and black-box cases. Especially, in the black-box case, the model was trained by extending the training dataset with additional synthetic data, and during the experiment, they found that the model, finally, overfits the synthetic data and could be one reason for privacy leakage.

Yeom *et al.* [113] and Leino *et al.* [114] have also investigated this problem. Yeom *et al.* [112] evaluated overfitting and privacy leakage quantitatively by targeting membership inference and attribute inference. The results on linear and tree models demonstrate that an attacker can sometimes perform better by only tuning the decision threshold at $\epsilon = \sigma_S$ instead of taking σ_D into consideration. They found that overfitting is a sufficient but not a necessary condition for membership inference since the model still can be attacked in a stable training algorithm (i.e., one that does not overfit), while attribute inference is more sensitive to overfitting.

For the second approach, Long *et al.* [106] analyzed the association between generalization and membership inference

and found that the success of membership inference may be caused by some unique influence due to the participation of different samples. In 2019, Nasr *et al.* [110] proposed a new framework for membership inference attack based on federated learning, considering overfitting in the NN. They pointed out that, in the later stages of federated learning, there is a phenomenon that the testing samples' gradients are always greater than that of the training samples. Therefore, they extend the work in [107] and show that the accuracy of the attack can reach 80% by taking the gradients in each layer of NN into consideration.

B. Unintended Feature Inference

In an unintended feature inference attack, the attacker's goal is to deduce features that are irrelevant to the trained prediction model. For example, a CNN for face recognition can leak some unintended features, such as wearing a pair of glasses. For a gender classification model, the attacker may be interested in some unintended features, such as the race information in the training set.

Melis *et al.* [114] proposed this interesting attack in collaborative learning and federated learning scenarios. To attack successfully, an attacker needs to generate a classifier by training with two kinds of data. One of them has the feature the attacker wants to infer, while the other does not. In the federated learning procedure, in each upload episode, the workers need to download the parameters and then upload the new parameters after training with their own dataset. The uploaded parameters for the above two different datasets will be different. Therefore, attackers can download the parameters for each round and then obtain two kinds of models: one that is trained by the data has the feature it wants to infer, whereas the other one does not. Collecting a certain number of different gradients and labeling them, the attacker can train a classifier to distinguish whether the parameters uploaded by a victim-user have the features or not. Although the attack has some limitations, it still performs successfully in some specific scenarios and remains an interesting threat to federated learning.

C. Representative Sample Reconstruction

In a representative sample reconstruction attack, the adversary's goal is to attack a certain federated learning worker by extracting the characteristics in the training sample. It can be seen as a shadow of the training sample, which looks similar and consists of the main representative information of the training sample, called the representative sample.

A representative sample reconstruction attack is generally performed through two approaches: one by applying GANs in the federated learning, and the other by completely reconstructing the trained model. This attack leads to direct privacy exploitation since the similarity of users' data can expose a lot of information. A representative sample consisting of characteristics of workers' data can be used to perform a poisoning attack [3], which could also substantially harm the global accuracy of federated learning. In 2017, Hitaj *et al.* [115] proposed the first representative sample reconstruction framework in federated learning.

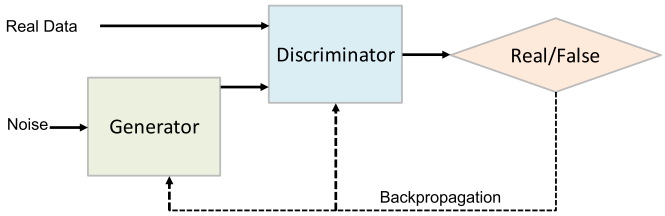


Fig. 5. Illustration of GANs.

Fig. 5 shows the generic way that GANs operate. It involves two models: one is *Discriminator* and the other is *Generator*. *Discriminator* is based on the discriminative model, which is trained based on sample data (real data), while *Generator* follows the generative model, which generates observation data based on some hidden information as input data. In GAN, *Generator* generates sample data based on a given noise vector, and then, *Discriminator* judges whether the input data are real or false. Two models are trained iteratively through backpropagation. The attacker can use the generated sample data, change its label to the desired (but wrong) class, add it to the training set of the global model, and, finally, train a model that can recognize the victim's dataset as wrong. The attacker also updates its parameter to the *global model* for combination. With successive rounds, the victim-user leaks more and more information about his dataset during training, and gradually, the *Generator* produces samples that are more similar to the victim-user's.

This idea is similar to the previous work on the model inversion for centralized model training [116] since both of them aim to recover the training data by extracting the characteristic information. Model inversion can be fooled by randomly generating some noise, which leads to classification error, i.e., recognizing a sample data, which does not match with the human-beings' perspective [117], [118]. Thus, model inversion can obtain sensitive information but also probably get meaningless information [115]. To avoid this problem, the representative sample reconstruction attack improves it with GANs [119], which not only ascertains the extraction of characteristics but also makes the information leaked by the generated data more meaningful (i.e., the data cannot be distinguished from the victim-user's training set easily).

Based on the work proposed by Hitaj *et al.* [115], Wang *et al.* [120] further extended this type of attack that is called *mGAN-AI*. Based on the idea that all users train the *Discriminator*, this GAN adds another procedure as follows. The algorithm catches the parameters updated by the victim-user to update the *Discriminator* as in the previous work. *mGAN-AI* further adds an additional step to generate some sample data through parameters by using L-BFGS algorithm [121] and then uses the data to train the *Discriminator* again for a more accurate recognition model. Finally, it interactively creates *Generator* with the help of the *Discriminator*. Though some prior knowledge of the user dataset is needed in *mGAN-AI*, this attack is much more accurate, and accordingly, the generated sample data is much more similar to the victim-user's original data.

D. Privacy-Preserving Frameworks

With the above possible privacy breaches, there is a concerted attempt to develop new privacy-preserving frameworks in federated learning. Currently, the solutions are mainly based on *differential privacy* [124], [125], *secure multiparty computation* [126], [127], and *homomorphic encryption* [128], [129]. In this section, we discuss these frameworks in detail.

Differential privacy adds noise to the information for privacy guarantee from a mathematical perspective [124]. Shorki *et al.* [130] applied to collaborative learning by letting each worker add noise to some of the parameters before uploading. This gives a much more promising privacy guarantee for selection upload with noise [107], [114]. However, this could lead to an extremely slow global convergence in small-scale federated learning systems [114]. Also, it may not work well in avoiding *representative sample reconstruction* attack [115].

Even with differential privacy, privacy leakage can occur in federated learning since the RA is able to observe the update proposed by each worker. One way to address this is by using *homomorphic encryption* of the parameters supplied by WAs to RA [131]. Homomorphic encryption encrypts gradient into a cipher text, and this cipher text has an operator equivalent to a certain operator of plain text. This enables computation on cipher text with the same result as with the plain text. For example, consider homomorphic encryption with the property

$$\text{Enc}(x_1 + x_2) = \text{Enc}(x_1) \circ \text{Enc}(x_2) \tag{15}$$

where the function $\text{Enc}(\cdot)$ represents an encryption to translate a plain text into a cipher text, x_1 and x_2 are two variables, and $\circ$ represent an operator for the cipher text. Homomorphic encryption can handle the problem of *curious but honest* RA, i.e., one that obeys the federated learning protocol honestly but may be curious about WA's data. However, such encryption is often quite expensive and must be designed for specific operations in mind. Also, since the gradient is not visible to the server, it is difficult to analyze the appropriateness of each user's parameter set. This could allow a poisoning attack possibly leading to a significant impact on global accuracy [3]. It also cannot deal with the collusion between RA and WAs. To address it, we need to introduce a trusted third party [132] or improve the algorithm with multikey homomorphic encryption algorithm proposed in [133].

Table III shows a summary of representative works in privacy attack technology.

VI. Discussion and Challenging Issues

A. Collaborative Learning Schemes

Although many collaborative learning schemes have been proposed for the full range of ML algorithms from unsupervised learning to supervised learning, the relative advantages of different types of SM decomposition and integration are still not investigated well. For example, while clustering, linear model, and SVM all benefit from IMI, they need to be further investigated in federated learning. Also, the scalability of distributed optimization methods in federated learning needs further study. We find that most of the collaborative learning schemes follow the passive and fixed participating model, which means that the root node mainly separates the learning tasks and distributes them to other helpers. Active and flexible participation schemes for collaborative learning largely remain to be investigated. The new learning scheme may bring new challenges with respect to model update coordination, energy consumption, optimization, and so on. For example, active participation requires new coordination mechanisms and their convergence analysis, which may be quite challenging.

To expand federated learning to mobile environments (e.g., training an autonomous driving model by aggregating inputs from many vehicles), we need to extend federated learning by jointly considering wireless channels, mobility model of users, and so on. For example, the iteration time slot and frequency should be optimized jointly based on mobile node characteristics. Furthermore, many other problems of such an environment, such as message/data drops, synchronization errors, and communication errors, need to be considered to ensure fast convergence and an accurate model. Finally, privacy-aware parameter compression needs significant additional work.

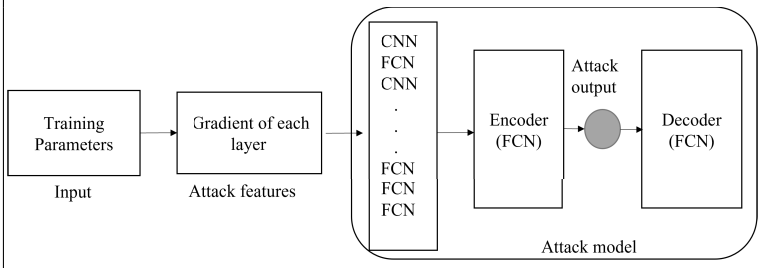
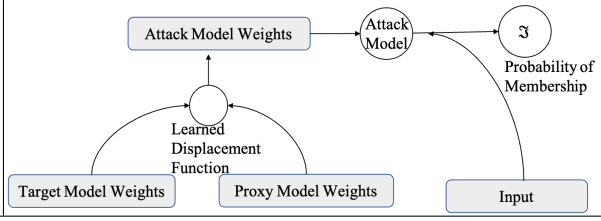
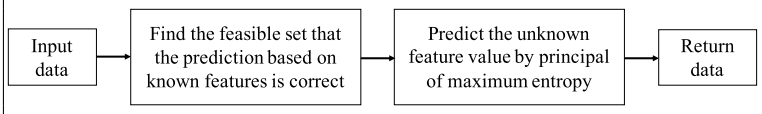
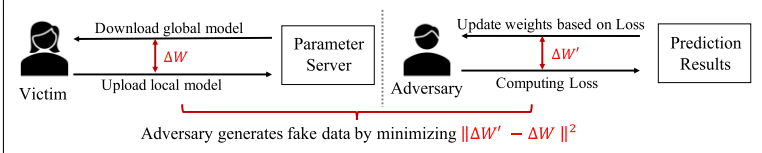
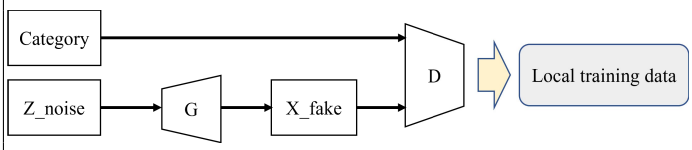
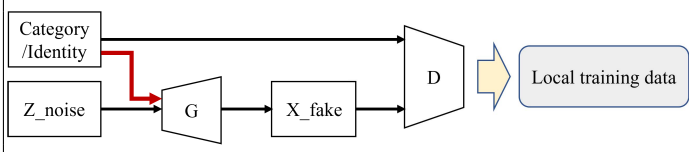
B. Robustness

By its very definition, a system is considered robust if it does not fail easily (i.e., in response to small or isolated perturbations). In the ML context, this amounts to ensure that the model does not provide an anomalous answer because of training on relatively small amounts of bad data with specific features. This can be addressed by training the model to be resistant to imperfections in the features, but this only provides a tradeoff between robustness and accuracy. In particular, a model trained on a wide range of imperfections to make it robust is likely to be less discriminative.

In the general case of an overall model consisting of multiple SMs, each SM needs to be made robust. This would require, at a minimum, that the RA validates each SM separately before fusing its outputs. This would still leave them exposed to the possibility of backdoors or perturbation of aspects not specifically covered by the validation tests. Covering such anomalies would require some redundancy, which can be provided in multiple ways. The simplest, but most costly, the mechanism is to hand out each SM for training to multiple WAs, exactly as in federated learning. For more sophisticated approaches, the decomposition of the model into SMs needs to be such that each feature is covered by multiple SMs. This can be challenging since the SM structure is often decided by the most relevant training data that a party possesses. Because of the complex relationship between the inputs and output of a DNN, it becomes difficult to compare SM outputs to determine vulnerabilities.

The general collaborative ML model shown in Fig. 3 points to a large number of scenarios, many of which can be very challenging to address. The most studied special case so far is federated learning, for which we have already discussed many types of attacks and some mitigation approaches. Nevertheless, addressing of attacks while preserving convergence and model accuracy remains challenging.

TABLE III
SUMMARY OF REPRESENTATIVE SAMPLE RECONSTRUCTION ATTACKS

Attack Types	Representative Works		Basic Procedure	Types	Results	Effects
Membership Inference Attack: measures a machine learning model and a record, determine whether this record was used as part of the model's training dataset or not	Shadow Model [108] [107]		As depicted in Fig. 4	Passive	Reduce the number of queries	Accuracy: 74.25%
	SGD-based [110]			Active	High attack capacities	Accuracy: 74.3%
	Over-fitting scenarios [111] [113] [111]			Active Passive	sufficient but not necessary	Accuracy [113] : 87.4 % CIFAR-100
Representative Sample Reconstruction: Attack a certain federated learning worker by extracting the characteristics in the training set, and generate a representative sample	Model Inversion [122] [116]			Passive	Significant blurry	IS: 1.01± 0.03
	Deep Leakage from Gradients [123]			Passive	Close to the real	MSE < 0.03 vs. previous > 0.2
	GAN	GAN-based [115]		Passive	Not Converge	IS: 1.18± 0.03
		mGAN-AI [120]		Passive Active	Close to the real one Most close to the real one	IS: 1.42± 0.03 1.61± 0.05

C. Privacy

As discussed above, privacy leakage occurs in many situations of collaborative learning and avoiding such a leakage often requires mechanisms that degrade the overall model or its convergence (e.g., adding noise to data) and may need computationally intensive operations, such as homomorphic encryption or secure multiparty computation discussed above. Privacy also has a direct impact on the ability to verify

things, such as ensuring that the training data or the returned parameters are not compromised. being able to simultaneously address the two opposing sides, namely, preserve privacy and yet detect/avoid malicious training or updates and do it in a cost-effective manner is a fundamental challenge that requires much further work.

D. Selection of Collaborative ML Algorithms

The selection of ML models is a common problem. Simple models, such as LR, DT, or SVM, have a simple and explainable formulation and are quite efficient to run. However, an NN can extract more complex features from data but at a much greater cost. Thus, collaborative LR/SVM/DT can be used in computation/communication constraint applications, such as sensor networks, since: 1) computation complexity of LM/SVM/DT is low; 2) the information to be transmitted is small (LR: Matrix X, SVM: subvectors, and DT: splitting information); and 3) most of them do not require interactive training. Collaborative NN and federated learning can be used for deep learning models, such as CNN and LSTM, and, thus, are applicable for image processing or language analysis tasks. In a collaborative scheme, the applications can be dealing with distributed hospital image data, autonomous driving data, audio data, and so on. The huge amounts of training data required by sophisticated models, such as LSTM, become a serious limiting factor to their use in many applications. Thus, the issue of newer, more easily trainable models, especially for data that incorporate some notion of the ordering of events remains an open problem.

E. Concluding Remarks

In this article, we have performed a comprehensive survey on collaborative learning, which is an important research topic with numerous applications. The survey covers most of the work that we are aware of in the corresponding research fields. We focused especially on different learning schemes, robustness, and privacy issues. The learning scheme is fundamental to collaborative learning and has implications for many other aspects, such as model update coordination, optimization, robustness, and privacy. We expect that, in the future, more general learning schemes, such as active or flexible participant schemes, may be investigated and will likely benefit many other application areas of collaborative learning. Robustness and privacy issues also need to be investigated much more deeply so that we can find ways of making collaborative ML truly robust in critical safety applications, such as automated driving. As to privacy, the existing work shows that overfitting affects privacy leakage differently for different types of attacks, e.g., membership inference or attribute inference. More theoretical and experimental studies are required for different types of privacy leakage. Finally, a study of the computation and communication overhead of privacy preservation schemes and techniques to make them more lightweight is crucial for their widespread use.

ACKNOWLEDGMENT

Parts of the work are an extension of the work under JST-NSF Joint Funding to study Big Data and Disaster (SICORP Project). The authors would like to thank Feiyuan Liang who is a master's student at Sun Yat-sen University for help with the literature review.

REFERENCES

[1] Q. Li *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," 2019, *arXiv:1907.09693*.
[2] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," 2018, *arXiv:1811.12470*.
[3] J. Zhang *et al.*, "Poisoning attack in federated learning using generative adversarial nets," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. (TrustCom/BigDataSE)*, Aug. 2019, pp. 374–380.
[4] S. Shen, S. Tople, and P. Saxena, "AUROR: Defending against poisoning attacks in collaborative deep learning systems," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, Dec. 2016, pp. 508–519.
[5] D. Cao *et al.*, "Understanding distributed poisoning attack in federated learning," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 233–239.
[6] D. Peteiro-Barral and B. Guijarro-Berdiñas, "A survey of methods for distributed machine learning," *Prog. Artif. Intell.*, vol. 2, no. 1, pp. 1–11, Mar. 2013.
[7] S. Devi, "A survey on distributed data mining and its trends," *Int. J. Res. Eng. Technol.*, vol. 2, no. 3, pp. 107–120, 2014.
[8] R. Gu *et al.*, "From server-based to client-based machine learning: A comprehensive survey," 2019, *arXiv:1909.08329*.
[9] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–43, Jul. 2020.
[10] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," 2019, *arXiv:1912.09789*.
[11] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.
[12] J. Wang *et al.*, "A field guide to federated optimization," 2021, *arXiv:2107.06917*.
[13] S. Ji, T. Saravirta, S. Pan, G. Long, and A. Walid, "Emerging trends in federated learning: From model fusion to federated x learning," 2021, *arXiv:2102.12920*.
[14] 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, pp. 1–19, 2019.
[15] J. Wang, M. C. Meyer, Y. Wu, and Y. Wang, "Maximum data-resolution efficiency for fog-computing supported spatial big data processing in disaster scenarios," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1826–1842, Aug. 2019.
[16] F. K. Dankar *et al.*, "Secure multi-party linear regression," in *Proc. EDBT/ICDT Workshops*, 2014, pp. 406–414.
[17] A. Gascón *et al.*, "Privacy-preserving distributed linear regression on high-dimensional data," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 4, pp. 345–364, 2017.
[18] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: Design and evaluation," *JMIR Med. Informat.*, vol. 6, no. 2, p. e19, Apr. 2018.
[19] H. P. Graf *et al.*, "Parallel support vector machines: The cascade SVM," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 521–528.
[20] C.-J. Hsieh *et al.*, "A divide-and-conquer solver for kernel support vector machines," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 566–574.
[21] E. Y. Chang, "PSVM: Parallelizing support vector machines on distributed computers," in *Foundations of Large-Scale Multimedia Information Management and Retrieval*. Springer, 2011, pp. 213–230.
[22] K. Flouri *et al.*, "Training a SVM-based classifier in distributed sensor networks," in *Proc. 14th Eur. Signal Process. Conf.*, Sep. 2006, pp. 1–5.
[23] B. Panda, J. S. Herbach, S. Basu, and R. J. Bayardo, "PLANET: Massively parallel learning of tree ensembles with MapReduce," *Proc. VLDB Endowment*, vol. 2, no. 2, pp. 1426–1437, Aug. 2009.
[24] A. Desai and S. Chaudhary, "Distributed decision tree," in *Proc. 9th Annu. ACM India Conf.*, Oct. 2016, pp. 929–934.
[25] T. Guo *et al.*, "Efficient distributed decision trees for robust regression," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Springer, 2016, pp. 79–95.
[26] F. Abuzaid *et al.*, "Yggdrasil: An optimized system for training deep decision trees at scale," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3817–3825.

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

[28] Y. Liu *et al.*, "Boosting privately: Federated extreme gradient boosting for mobile crowdsensing," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Nov. 2020, pp. 1–11.

[29] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 4642–4649.

[30] J. Dass, V. Sarin, and R. N. Mahapatra, "Fast and communication-efficient algorithm for distributed support vector machine training," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1065–1076, May 2019.

[31] J. Dass, V. N. S. P. Sakuru, V. Sarin, and R. N. Mahapatra, "Distributed QR decomposition framework for training support vector machines," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 753–763.

[32] W. Kim, M. S. Stanković, K. H. Johansson, and H. J. Kim, "A distributed support vector machine learning over wireless sensor networks," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2599–2611, Nov. 2015.

[33] W. Fang, B. Yang, D. Song, and Z. Tang, "A new scheme on privacy-preserving distributed decision-tree mining," in *Proc. 1st Int. Workshop Educ. Technol. Comput. Sci.*, 2009, pp. 517–520.

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

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

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

[37] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*.

[38] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," 2018, *arXiv:1802.07876*.

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

[40] M. Mohri, G. Sivek, and A. Theertha Suresh, "Agnostic federated learning," 2019, *arXiv:1902.00146*.

[41] Y. Jiang, J. Konečný, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," 2019, *arXiv:1909.12488*.

[42] E. Januzaj *et al.*, "DBDC: Density based distributed clustering," in *Proc. Int. Conf. Extending Database Technol.* Springer, 2004, pp. 88–105.

[43] E. Januzaj *et al.*, "Scalable density-based distributed clustering," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discovery*. Springer, 2004, pp. 231–244.

[44] X. Xu *et al.*, "A fast parallel clustering algorithm for large spatial databases," *Data Mining Knowl. Discovery*, vol. 3, no. 3, p. 27, 1999.

[45] Y. He *et al.*, "MR-DBSCAN: An efficient parallel density-based clustering algorithm using MapReduce," in *Proc. IEEE 17th Int. Conf. Parallel Distrib. Syst.*, Dec. 2011, pp. 473–480.

[46] M. Bendechache and M.-T. Kechadi, "Distributed clustering algorithm for spatial data mining," in *Proc. 2nd IEEE Int. Conf. Spatial Data Mining Geographical Knowl. Services (ICSDM)*, Jul. 2015, pp. 60–65.

[47] J.-F. Laloux *et al.*, "Efficient distributed approach for density-based clustering," in *Proc. 20th IEEE Int. Workshops Enabling Technol., Infrastructure Collaborative Enterprises, (WETICE)*, Jun. 2011, pp. 145–150.

[48] M. N. Joshi, "Parallel K-means algorithm on distributed memory multiprocessors," *Computer*, vol. 9, p. 12, 2003.

[49] S. Merugu and J. Ghosh, "Privacy-preserving distributed clustering using generative models," in *Proc. 3rd IEEE Int. Conf. Data Mining*, Nov. 2003, pp. 211–218.

[50] X. Li *et al.*, "On the convergence of FedAvg on non-IID data," in *Proc. ICLR*, 2020, pp. 1–26.

[51] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. 2017, pp. 1273–1282.

[52] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[53] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 1205–1221, Jun. 2019.

[54] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.

[55] 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.

[56] F. Sattler *et al.*, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 1–14, Sep. 2019.

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

[58] 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, *arXiv:1811.11479*.

[59] T. Sakai *et al.*, "Parallel processing for density-based spatial clustering algorithm using complex grid partitioning and its performance evaluation," in *Proc. Int. Conf. Parallel Distrib. Process. Techn. Appl. (PDPTA)*, 2016, p. 337.

[60] C. Böhm, R. Noll, C. Plant, and B. Wackersreuther, "Density-based clustering using graphics processors," in *Proc. 18th ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2009, pp. 661–670.

[61] B. Welton *et al.*, "Data reduction and partitioning in an extreme scale GPU-based clustering algorithm," in *Proc. 2nd Int. Workshop Data Reductions Big Sci. Data (DRBSD)*, Denver, CO, USA, 2017.

[62] B. Welton, E. Samanas, and B. P. Miller, "Mr. Scan: Extreme scale density-based clustering using a tree-based network of GPGPU nodes," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2013, pp. 1–11.

[63] J. George and P. Gurram, "Distributed stochastic gradient descent with event-triggered communication," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 5, Apr. 2020, pp. 7169–7178.

[64] Y. Fan, G. Feng, Y. Wang, and C. Song, "Distributed event-triggered control of multi-agent systems with combinational measurements," *Automatica*, vol. 49, no. 2, pp. 671–675, Feb. 2013.

[65] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 3, no. 1, pp. 1–130, 2009.

[66] X. Chang, S.-B. Lin, and D.-X. Zhou, "Distributed semi-supervised learning with kernel ridge regression," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 1493–1514, Jan. 2017.

[67] J. Bilmes *et al.*, *Parallel Graph-Based Semi-Supervised Learning*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[68] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," 2017, *arXiv:1705.10467*.

[69] Y. Zhang and Q. Yang, "A survey on multi-task learning," 2017, *arXiv:1707.08114*.

[70] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: A survey and extensions," *Math. Methods Oper. Res.*, vol. 66, no. 3, pp. 373–407, Dec. 2007.

[71] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[72] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Comput. Sci. Rev.*, vol. 34, Nov. 2019, Art. no. 100199.

[73] B. Biggio, L. Didaci, G. Fumera, and F. Roli, "Poisoning attacks to compromise face templates," in *Proc. Int. Conf. Biometrics (ICB)*, 2013, pp. 1–7.

[74] B. Biggio, L. Didaci, G. Fumera, and F. Roli, "Poisoning attacks to compromise face templates," in *Proc. Int. Conf. Biometrics (ICB)*, Jun. 2013, pp. 1–7.

[75] B. Nelson and A. D. Joseph, "Bounding an attack's complexity for a simple learning model," in *Proc. 1st Workshop Tackling Comput. Syst. Problems Mach. Learn. Techn. (SysML)*, Saint-Malo, France, 2006, p. 111.

[76] M. Kloft *et al.*, "Online anomaly detection under adversarial impact," in *Proc. AISTATS*, 2010, pp. 405–412.

[77] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. ICLR*, 2014.

[78] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, Y. Bengio, Eds., 2015.

[79] B. Biggio *et al.*, "Poisoning attacks against support vector machines," in *Proc. ICML*, 2012.

[80] H. Xiao *et al.*, "Adversarial label flips attack on support vector machines," in *Proc. ECAI*, 2012, pp. 870–875.

[81] X. Chen *et al.*, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*.

[82] C. Yang *et al.*, "Generative poisoning attack method against neural networks," 2017, *arXiv:1703.01340*.

[83] L. MuñozGonzález *et al.*, "Towards poisoning of deep learning algorithms with back-gradient optimization," 2017, *arXiv:1708.08689*.

[84] E. Bagdasaryan *et al.*, "How to backdoor federated learning," 2018, *arXiv:1807.00459*.

[85] J. Hayes *et al.*, "Contamination attacks and mitigation in multi-party machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6604–6615.

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

[87] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Comput. Sci. Rev.*, vol. 34, Nov. 2019, Art. no. 100199.

[88] N. Carlini *et al.*, "Adversarial examples are not easily detected: Bypassing ten detection methods," 2017, *arXiv:1705.07263*.

[89] A. S. Chivukula and W. Liu, "Adversarial learning games with deep learning models," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 2758–2767.

[90] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[91] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," 2016, *arXiv:1605.07277*.

[92] Y. Dong *et al.*, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9185–9193.

[93] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1625–1634.

[94] N. Papernot *et al.*, "Distillation as a defense to adversarial perturbations against deep neural networks," 2015, *arXiv:1511.04508*.

[95] N. Carlini *et al.*, "Towards evaluating the robustness of neural networks," 2016, *arXiv:1608.04644*.

[96] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.

[97] J. Wang, "Adversarial examples in physical world," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1–2.

[98] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," 2017, *arXiv:1708.06131*.

[99] J. Hayes *et al.*, "Learning universal adversarial perturbations with generative models," in *Proc. IEEE SP Workshops*, May 2018, pp. 43–49.

[100] A. Athalye *et al.*, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. ICML*, 2018, pp. 274–283.

[101] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," in *Proc. ICLR*, 2018, pp. 1–28.

[102] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.

[103] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant SGD," 2018, *arXiv:1802.10116*.

[104] P. Blanchard *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 119–129.

[105] L. Li *et al.*, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1544–1551.

[106] Y. Long *et al.*, "Understanding membership inferences on well-generalized learning models," 2018, *arXiv:1802.04889*.

[107] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[108] H. Yu, K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin, "CloudLeak: Large-scale deep learning models stealing through adversarial examples," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–16.

[109] F. Tramèr *et al.*, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Secur. Symp. (USENIX Security)*, 2016, pp. 601–618.

[110] 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," 2018, *arXiv:1812.00910*.

[111] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 587–601.

[112] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Found. Symp. (CSF)*, Jul. 2018, pp. 268–282.

[113] K. Leino *et al.*, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in *Proc. USENIX Secur. Symp.*, 2020, pp. 1605–1622.

[114] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 691–706.

[115] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 603–618.

[116] 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. Secur.*, Oct. 2015, pp. 1322–1333, doi: 10.1145/2810103.2813677.

[117] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[118] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[119] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, vol. 2. Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680.

[120] 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 Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2512–2520.

[121] *Wikipedia: Limited-Memory BFGS Algorithm*. [Online]. Available: https://en.wikipedia.org/wiki/Limited-memory_BFGS

[122] M. Fredrikson *et al.*, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proc. 23rd USENIX Secur. Symp. (USENIX Security)*, 2014, pp. 17–32.

[123] L. Zhu *et al.*, "Deep leakage from gradients," in *Federated Learning*. Springer, 2020, pp. 17–31.

[124] C. Dwork *et al.*, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology*. 2006.

[125] C. Dwork *et al.*, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014, doi: 10.1561/0400000042.

[126] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. IEEE Symp. Found. Comput. Sci. (FOCS)*, Oct. 1986, pp. 162–167.

[127] M. Yung, "From mental poker to core business: Why and how to deploy secure computation protocols?" in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2015, pp. 1–2, doi: 10.1145/2810103.2812701.

[128] *Sciencedirect: Homomorphic Encryption*. Accessed: May 20, 2022. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/homomorphic-encryption

[129] P. V. Parmar, S. B. Padhar, S. N. Patel, N. I. Bhatt, and R. H. Jhaveri, "Survey of various homomorphic encryption algorithms and schemes," *Int. J. Comput. Appl.*, vol. 91, no. 8, pp. 26–32, Apr. 2014.

[130] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 1310–1321.

[131] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

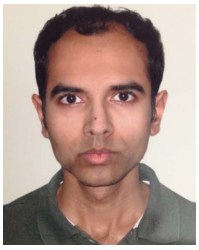
[132] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[133] P. Li *et al.*, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generat. Comput. Syst.* vol. 74, pp. 76–85, Sep. 2017.

Junbo Wang (Member, IEEE) received the B.S. and M.S. degrees in electrical and electronic engineering from Yanshan University, Qinhuangdao, China, in 2004 and 2007, respectively, and the Ph.D. degree in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan, in 2011.

He was a Post-Doctoral Scholar and an Associate Professor with The University of Aizu. He is currently an Associate Professor with the School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen, China. His research interests include collaborative machine learning (ML), federated learning, fog computing, big data, and privacy.

Dr. Wang was a Project Leader (PI) at the Japan site for JST-NSF Joint Funding to study Big Data and Disaster (SICORP Project).



Amitangshu Pal (Member, IEEE) received the Ph.D. degree from the Electrical and Computer Engineering (ECE) Department, The University of North Carolina at Charlotte, Charlotte, NC, USA, in 2013.

He is currently an Assistant Professor with IIT Kanpur, Kanpur, India. He has published over 70 conferences papers and journal articles. His current research interests include wireless sensor networks, reconfigurable optical networks, smart healthcare, cyber–physical systems, mobile and pervasive computing, and cellular networks.

Qinglin Yang (Member, IEEE) received the B.S. degree from the School of Geographical Sciences, Guangzhou University, Guangzhou, China, in 2014, the M.S. degree in computing mechanism from the College of Civil Engineering, Kunming University of Science and Technology, Kunming, China, in 2017, and the Ph.D. degree in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan, in 2021.

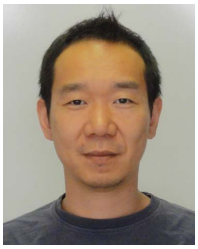
He currently holds a post-doctoral position at the School of Intelligent System Engineering, Sun Yat-sen University, Shenzhen, China. His research interests include edge computing, federated learning, and green computing.

Krishna Kant (Life Fellow, IEEE) received the Ph.D. degree in mathematical sciences from The University of Texas at Dallas, Richardson, TX, USA, in 1981.

He was a Research Professor with the Center for Secure Information Systems, George Mason University, Fairfax, VA, USA. He has served in the industry for 18 years (at Intel, Bellcore, and Bell Labs) and ten years in academia (at Penn State University, State College, PA, USA, and Northwestern University, Evanston, IL, USA). From 2008 to 2013, he was the Program Director of NSF where he managed the Computer Systems Research Program and was instrumental in the development and running of the NSF-wide sustainability initiative named Science, Engineering and Education for Sustainability (SEES). He carries a combined 41 years of experience in academia, industry, and government. He is currently a Professor with the Computer and Information Science Department, Temple University, Philadelphia, PA, USA, where he directs the IUCRC Center on Intelligent Storage. He has published in a wide variety of areas in computer science and authored a graduate textbook on the performance modeling of computer systems. His research interests span a wide range, including data center storage and networking, communications in challenging environments, and robustness and security in cyber and cyber–physical systems.

Kaiming Zhu received the B.E. degree in computer science and engineering from Northeastern University, Shenyang, China, in 2019. He is currently pursuing the master's degree with Sun Yat-sen University, Shenzhen, China.

His research interests include federated learning and privacy-preserving mechanism.



Song Guo (Fellow, IEEE) is currently a Full Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. He published many papers in top venues with a wide impact on these areas and was recognized as a Highly Cited Researcher (Clarivate Web of Science). His research interests are mainly in big data, edge AI, mobile computing, and distributed systems.

Prof. Guo was a member of the IEEE ComSoc Board of Governors. He is also a fellow of the Canadian Academy of Engineering and the IEEE Computer Society. He was a recipient of over a dozen best paper awards from IEEE/ACM conferences, journals, and technical committees. He is also the Editor-in-Chief of IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY and the Chair of the IEEE Communications Society (ComSoc) Space and Satellite Communications Technical Committee. He has served as the chair of organizing and technical committees of many international conferences. He was an IEEE ComSoc Distinguished Lecturer. He has served the IEEE Computer Society on the Fellow Evaluation Committee and has been named on the editorial board of a number of prestigious international journals, such as IEEE TPDS, IEEE TCC, and IEEE TETC.