Collaborative Machine Learning: Schemes, Robustness and Privacy

Junbo Wang *Member, IEEE,* Amitangshu Pal, *Member, IEEE,* Kaiming Zhu, *NonMember, IEEE,* Krishna Kant, *Fellow, IEEE,* Wuhui Chen, *Member, IEEE,* Song Guo, *Fellow, IEEE,*

Abstract—Distributed Machine Learning was originally investigated to solve a complex machine learning problem in a parallel way for more efficient usage of computation resources. It also meets the requirements when the data are generated in a distributed way, such as an edge computing environment, e.g., intelligent connected vehicles. In recent years, federated learning was further proposed to perform training with a shared learning model, and parameter aggregation and privacy-preserving schemes. However, there are still a lot of challenging issues in the collaborative machine learning. Some of them have been already considered by researchers, and most of them are still open problems. To present a more clear technical road-map of the collaborative machine learning, we have performed a comprehensive survey in this paper. The survey covers a broad of machine learning schemes in the collaborative scenarios, including collaborative clustering, collaborative support vector machine, and federated learning (collaborative neural network). Meanwhile, since collaborative learning is based on parameter exchanges during learning process, which brings privacy leakage possibility and robustness issues in learning. In this survey, we especially pay attention on the current state of arts of privacy and robustness in collaborative machine learning.

Index Terms—Collaborative Learning, Distributed Learning, Federated Learning, Privacy, Robustness.

1 Introduction

Machine Learning (ML) has proved successful in numerous applications including object recognition, autonomous driving, stock prediction, and so on. Collaborative ML involves multiple computing nodes in order to increase computing resources, avoid unnecessary transfer of large training datasets, and conduct training on datasets without having to reveal them. Collaborative ML works as follows: raw data are first collected at local sites and used to generate local models (or submodels), which can be support vectors, clusters, weight matrix in a neural network. The submodels are then aggregated at a root node based on the nature of the submodels and the aggregation approach.

Collaborative ML algorithms can be based on a variety of learning schemes, which makes the aggregation models challenging and ad hoc. For example, the support vector machine (SVM) formalizes a problem to maximize soft margin between support vectors, and can be transformed as a quadratic programming (QP) problem. While a neural network generates a non-linear model, the model is often optimized by gradient descent. There are a lot of aggregation models and optimization methods designed separately for each kind of collaborative learning method. They are sometimes correlated and most of time designed separately. However there needs a comprehensive survey covering various types of collaborative learning models.

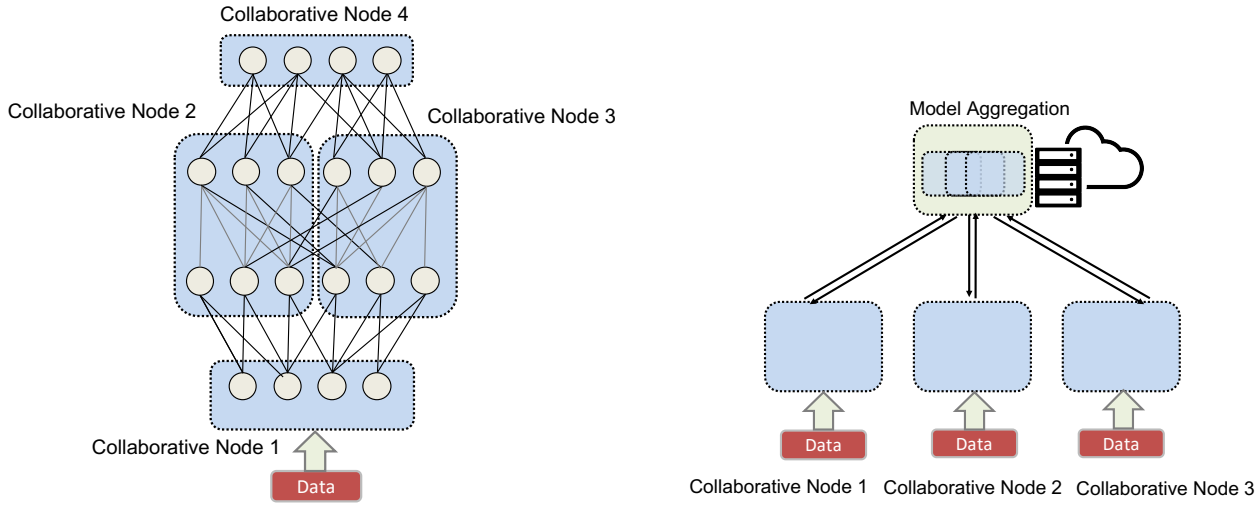
Several survey works have been performed trying to summarize the related technologies on collaborative learning [1] [2] [3] [4] [5] [6] [7]. Reference [1] summarises approaches for combining predictions of a set of classifiers, including decision rules, stacked generalization, and so on. In [2], the methods related on collaborative classifier learning, collaborative association rule mining, collaborative clustering etc. are briefly summarized. A recent work in [3] presents a comparison of distributed ML on mobile devices. Reference [4] focuses on distributed deep neural network, reference [5] explores communication structures and optimization for distributed learning, and [7] describes vertical and horizontal federated learning. Comparing with the above surveys, our work pays more attentions on collaborative learning schemes, and robustness, privacy issues during the collaborative procedure.

While collaborative learning 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 [6]. Although these issues have been discussed in the literature under various assumptions [8], [9], [10], [11], there is no integrated methodology to consider various forms of robustness challenges.

The purpose of this paper is to address these gaps. We cover most of the popular collaborative ML algorithms include collaborative clustering, collaborative SVM, collaborative decision tree and collaborative neural network (NN) mode. We further discuss the type of aggregation in each collaborative learning algorithm. On robustness, we provide a taxonomy of various types of attacks, including data poisoning attacks, model poisoning attacks, and black-box, grey-box white-box attacks. Then we further discuss and compare possible privacy-leakage and the corresponding

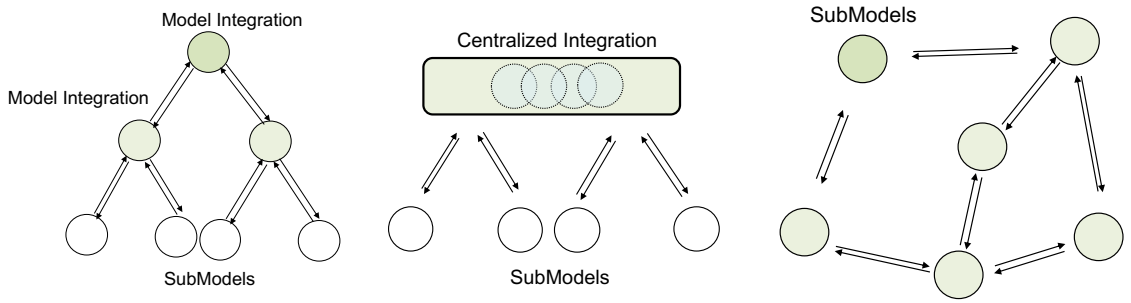
• *Junbo Wang is with Guangdong Provincial Key Laboratory of Intelligent Transportation System, School of Intelligent Systems Engineering, Sun Yat-Sen University, Shenzhen, China.*
E-mail: j-wang@ieee.org

Manuscript received April 19, 2005; revised September 17, 2014.



(a) Model Parallelism (b) Data Parallelism

Fig. 1. Two Types of Parallelism for Collaborative ML.



(a) Incremental Model Integration (b) Centralized Model Integration (c) Fully Distributed Model Integration

Fig. 2. Three Types of Model Integration.

protection methods.

The structure of the rest of the paper is as follows: section 2 gives a brief introduction to collaborative learning and its applications; section 3 summarizes collaborative unsupervised learning techniques, and section 4 compares collaborative supervised techniques. Sections 5 and 6 then summarize research on robustness and privacy related issues, respectively. Finally, section 7 discusses future challenges and section 8 concludes the paper.

We first start this survey from the basic schemes for collaborative ML and its applications in the following section.

2 Collaborative Learning and its Applications

2.1 Distributed/parallel Learning

Distributed and parallel learning was originally 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 sub-parts, 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 almost full range of unsupervised learning and supervised learning as shown in Table 1. In particular, it covers all the three types of model integration, illustrated in Fig. 2. These are: (a) Incremental Model Integration (IMI), where submodels are integrated hierarchically, (b) Centralized Model Integration (CMI), which represents one-step integration of all submodels, and (c) Fully Distributed Model Integration (FDMI), where the integration follows some general acyclic graph, meaning that certain submodels may be integrated along multiple paths.

2.2 Federated Learning

Recently, more and more researches focus on Federated Learning, which is based on a slave-master learning model and more suitable for Neural Network.

Inspired by the collaborative scenario of ML, two research scientists from Google proposed a new learning framework in Google AI Blog 2017 [12], which is called *Federated Learning* and illustrated in Fig. 3. The main assumption of federated learning is that each worker agrees on the same network model structure (neural network) which can undergo local training and global aggregation. For local training, each participating worker agent downloads the

TABLE 1
Different Collaborative Learning

Type of Collaborative Learning	Model Integration	Unsupervised Learning	Supervised Learning			
		Clustering	Linear Model	SVM	Decision Tree	Neural Network
Distributed Learning	IMI, CMI, FDMI	Yes	Yes	Yes	Yes	Yes
Federated Learning	Mainly CMI	/	Yes	/	Yes	Yes

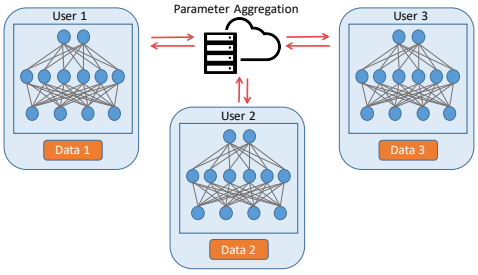


Fig. 3. Structure of Federated Learning

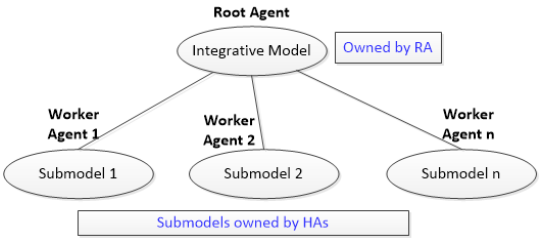


Fig. 4. Illustration of Collaborative Learning

parameters from the global (or root) server, initializes the model, and performs model training with local data, until certain accuracy is achieved. Then, each worker updates its training results (e.g., weights in the neural network) to the root server. The root server, which is chosen to do aggregation of parameters, computes updates to the parameters for the next round of local training using certain aggregation algorithm (e.g., averaging parameters from all agents).

2.3 Collaborative Learning

Distributed learning and federated learning target similar research topics, however their coverage is different based on our survey. Distributed learning covers most of ML schemes, and while federated learning focuses on neural networks mostly, even if similar functionality can be implemented using a linear model and decision tree. Federated learning studies typically consider privacy-preservation issues during the collaborative learning procedure among distributed nodes. To represent the integration of distributed learning and federated learning, in this survey we use terminology "collaborative learning" during the discussions.

Here we assume a much more general model of collaborative learning involving a root agent (RA) seeking the help of a number of worker agents (WAs) in performing its task as illustrated in Fig. 4. The underlying assumption is that each HA is an independent party and has its own computing and storage resources to help with the training and/or inferencing. In particular, each HA works with a submodel, which may be either a part of the larger model or the entire model itself.

In the simplest case, the submodel may be the entire model – this corresponds to the federated learning situation, where the purpose of the distribution is to train the model on different data sets collaboratively without any HA having to reveal its dataset. More generally, the RA may partition the model into multiple submodels 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 submodels using a higher level model, which could range from a simple aggregation of submodel results to being inputs to another ML model run by RA. For instance, different submodels may focus on different features or learning aspects which are then integrated by the RL. A concrete example of this is where each submodel recognizes a separate language in a multi-lingual 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 HA'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 HA's for further training.

2.4 Possible Applications

There are quit a lot of possible applications on collaborative ML.

2.4.1 Edge Computing Empowered Applications

Collaborative ML is well suited for edge computing empowered applications because of the possibility of training submodels at edge nodes in a privacy preserving way, and possibly doing part of the inferencing at edge nodes for quicker responses. Intelligent Connected Vehicle (ICV) is one typical application for Edge Computing Empowered Intelligence, where they can improve their situation awareness by communicating with other nearby vehicles and with the roadside unit (RSU). For example, the camera or LiDAR (Light Detection and Ranging) collect the data around the

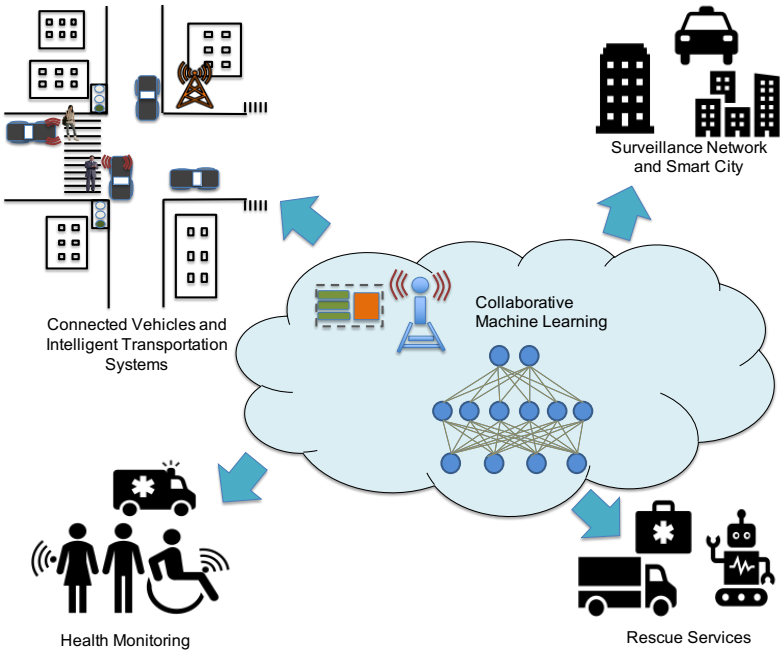


Fig. 5. Applications with Edge Intelligence Empowered by collaborative ML

vehicle, and then local data processing can better detect and recognize objects on the road. Path planning [13] is another capability well suited for the edge, and it concerns finding the optimum path by analyzing the surrounding information obtained from other vehicles [14]. Path planning typically uses deep reinforcement learning and deep neural network, both of which can be quite heavy duty and thus benefit from collaboration.

2.4.2 Health Monitoring

As the personalized health monitoring devices proliferate, the data collected by them can be used to train federated models for recognizing specific health risks [15] without having to reveal the data or send it to a central entity. Such devices can monitor one or more of important parameters such as heart rate, blood pressure, body temperature, drowsiness, posture, breathing, etc. [16], and most retain history for charting purposes. Such devices can pull in personal health care records from clinics and hospitals and thus can provide a rich source of data to train models without information leakage.

2.4.3 Disaster Response

Natural disasters often lead to an inadequate, unstable or overloaded communications infrastructure for multiple reasons including physical damage to network or power and high demand for communications. The computing/communications hubs deployed to augment the network capacity and coordinate rescue operations can be used for training models based on locally collected data without putting much stress on the overwhelmed network. In our previous research [17], we have shown that the collaborative spatial clustering can save data transmission time among different areas, and yet achieve acceptable data analysis accuracy.

3 Collaborative Unsupervised Learning

The most popular unsupervised learning is clustering. Collaborative clustering works as follows: for each distributed data processing node, it collects raw data from the surrounding sensing devices and then performs clustering. The procedure extracts representative points to generate several local clusters for representation of the local data based on different policies. The local clusters are then aggregated in a root node.

Take DBSCAN as an instance [18], it is used to find groups of points satisfying a specific condition, and is well studied for distributed scheme based on density. In this paper, we use DBSCAN to stimulate the discussion, and other types of collaborative can be found in the Table 2. A cluster from DBSCAN contains some minimum number of objects ($MinPts$) within a circle with given radius (Eps). The objects are selected as follows:

Directly density-reachable: An object A is directly density-reachable from another object B if (1) A belongs to neighbor objects of B within a radius Eps, and (2) the number of neighbor objects of B is greater than a predefined value $MinPts$.

Density-reachable: An object A is density-reachable to another object B, by a chain of objects $O_1, ..., O_n$, where $O_1 = A, O_n = B$ if every O_{i+1} is directly density-reachable from O_i.

Then DBSCAN starts with an arbitrary core point A and retrieves all points density-reachable from A to generate a cluster.

3.1 Approaches for Collaborative Clustering

DBSCAN has been extended work in a distributed/parallel way. In [19], DBDC (Density-Based Distributed Clustering) proposes a parallel implementation of DBSCAN. It first

clusters the data locally, and then it extracts aggregated information about the locally created clusters and sends this information to a central (root) node. At the root node, a global cluster is reconstructed based on the aggregated information. PartDBSCAN [20] uses a master-slave model and adopts a dR-tree to balance the partition of data over different slave nodes. The data in slave nodes are clustered and separated, then results from the different slave nodes are merged together to make a global cluster. Such an approach can also be applied into other types of clustering, e.g., grid-based clustering in [17].

Collaborative clustering has also been developed to work in an edge computing environment as shown in [17], where the raw data from each edge node is partially processed locally and then local clusters and unprocessed raw data are transmitted to its parent node. Data aggregation is further performed at each edge node to integrate raw data and the local clusters sent from its child node together to generate a new local cluster, as shown in Fig. 6. An important research issue in collaborative clustering is to select suitable policy to represent local cluster. Better selections can achieve higher cluster accuracy while still reducing the transmission data size as discussed next.

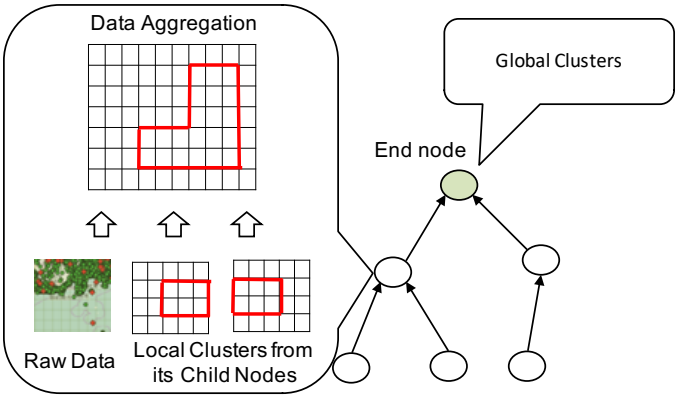


Fig. 6. Collaborative Clustering in Edge-Computing Environment

3.2 Policies for Representing Local Clusters

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 [20] [21] [22] [23], which can be seen as core points in a region. For example, in DBSCAN, core points are selected to represent a cluster if in a given radius (*EPs*), at least a minimum number of points (MinPts) in the cluster.

Specific core-points are the points selected from the core points to further reduce the size of transmission data in collaborative spatial clustering methods. In DBDC [19], the $Scor_C$ 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 [24] [25]. 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 result 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 [26], [27]. 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 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.

4 COLLABORATIVE SUPERVISED LEARNING

Collaborative learning is more broadly investigated in the context supervised (rather than unsupervised) learning. In the following we discuss the popular collaborative schemes including linear models, support vector machine, decision tree and neural network.

4.1 Collaborative Linear Models

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

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

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

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

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

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3}$$

Current multi-party collaborative linear regression [28], [29] mainly collect $(\mathbf{X}^T\mathbf{X})^{-1}$ from the collaborating nodes, and then aggregate them in a root node. The aggregation can also be done in a distributed manner when adopting gradient descent algorithm in ML [30]. In theory, *Secure multiparty computation* can be used for privacy preservation in data sharing, and *homomorphic encryption* can be used to retain confidentiality of the data passed to collaborating nodes; however, the practicality of these rather heavy-duty mechanisms remains questionable.

4.2 Collaborative Support Vector Machine

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

4.2.1 Foundation of Support Vector Machine (SVM)

The Support Vector Machine 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 $\mathcal{X} = \{\mathbf{x}_i, y_i)|\mathrm{x}_i \in \mathbb{R}^d\}$, where $\mathbf{x}_i$ is a feature vector for training the learning model and y_i's are the labels, the problem can be formalised as follows,

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

where b is the threshold, $\mathbf{w}$ is a weight vector, and C is a regularization hyperparameter that determines the trade-off 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,

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

where $Q_{ij} = y_i y_j x_i x_j$, and α is a vector of Lagrangian multiplier. The weight vector $\mathbf{w}$ has relation with α that $\mathbf{w} = \sum_{i=1}^{n}\alpha_i x_i$. Then the problem can be solved by solvers for QP problem or Sequential Minimal Optimization (SMO) which is more efficient with small size of α 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. Type 2 schemes are used for dealing with data generated from distributed sensing devices. We discuss these in the following.

4.2.2 Type 1: Parallel Designs of Centralized SVM

The basic idea of Type 1 is to separate the whole data set and the problem into several subsets and subproblems, then solve the subproblems separately, and finally integrate the sub-results together. The purpose is to decrease computation or space complexity to solve the QP problem in Eq. (5).

Cascade SVM [31]: Cascade SVM is a kind of multi-layer approach. In the first layer, the data are split into subsets and each one is evaluated individually for support vectors. The results are combined two-by-two as a kind of "reduction tree" for the next layer. The resulting support vectors from the last layer are transferred into the first layer, treated together with the non-support vectors. It is proved that it can converge to the global optimum because it keeps the best set of support vectors produced in one layer, and uses it in at least one of the subsets in the next layer in Cascade architecture. For Cascade, only support vectors generated in the current layer will be passed to the next layer [32].

Divide-and-Conquer SVM (DC-SVM) [33]: Similar with Cascade SVM, DC-SVM also divides the whole data into several small subsets and solves them independently. More specifically, in divide step, DC-SVM divides the full problem into smaller subproblems, which can be solved independently and efficiently. For example, by dividing the whole problem into k subproblems with equal sizes, the time complexity for solving the subproblems can be dramatically reduced to $\mathcal{O}(n^2/k)$, while the computation complexity of the original QP problem is at least $\mathcal{O}(n^2)$. Meanwhile, the space complexity is reduced to $\mathcal{O}(n^2/k^2)$ from $\mathcal{O}(n^2)$. After computing all the subproblems, an approximate solution for the whole problem is formed as $\overline{\alpha} = [\overline{\alpha}_1, \overline{\alpha}_2, ..., \overline{\alpha}_k]$. The paper shows that the difference from the exact solution α^* is bounded as follows:

$$0 \leq f(\overline{\alpha}) - f(\alpha^*) \leq (1/2)C^2 D(\psi) \tag{6}$$

where $f(.)$ represent the objective function in Eq. (5), and $D(\psi)$ is calculated as follows

$$D(\psi) = \sum_{i,j:\psi(x_i)\neq\psi(x_j)} |K(x_i, x_j)| \tag{7}$$

where $\psi(x_i)$ denotes the subset to which x_i belongs. From Eq. (7) we can see that the accuracy of DC-SVM depends on the number of subsets k and how the data is divided among them. A smaller k yields better accuracy but needs more computation, since the computation complexity is $\mathcal{O}(n^2/k)$.

Parallelizing SVM on Distributed Computers (PSVM) [34]: PSVM further reduces memory use through a matrix factorization. It loads only essential data in each machine to perform parallel computation. Given n training instances with d dimensions, PSVM first separates data and loads the training data onto m machines. Next it performs a parallel row-based Incomplete Cholesky Factorization (ICF) on the loaded data, and stores only factorized matrix with p dimension in each machine. Finally, PSVM performs parallel Interior-Point method to solve the quadratic optimization problem in Eq. (5).

PSVM reduces the memory requirement from $\mathcal{O}(n^2)$ to $\mathcal{O}(np/m)$, and decreases computation time to $\mathcal{O}(np^2/m)$. The prediction accuracy is evaluated experimentally by setting p of PSVM to n^t, and t is from 0.1 to 0.5, while the accuracy can be close to 1 when setting $t = 0.5$. The accuracy becomes poor with decreasing of t. However, PSVM uses kernel matrix approximation based on ICF, which makes it unfit for large datasets [35].

To exploit the potential of faster convergence, in recent works [36] [35], QRSVM framework was proposed by implementing QR decomposition into collaborative SVM. They use kernel approximation technique, and present the low-rank approximation of the kernel matrix in a separable form for fitting to the distributed framework. They further decompose the approximated kernel matrix using QR factorization to achieve memory efficient representations. The evaluation in [36] shows that QRSVM can converge in around 2 minutes using 16 processors, while PSVM takes 20 minutes in the same setup.

4.2.3 Type 2: Collaborative SVM for Distributed Data

Type 2 considers a scenario that there are a lot of computation nodes distributed in a specific area. They have abilities of both collecting data and training on local data to generate immediate data (e.g., local support vectors). Several approaches are relevant in this case:

Incremental approach: In [37] the authors 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 are 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.

Fully-distributed Scenario: In [38], the authors employ the concept of gossip-based incremental SVM with a geometric representation. Through the join operation of convex hulls from different distributed nodes, the proposed algorithm can guarantee the convergence in a finite time to the global solution. Reference [39] uses alternating direction method of multipliers (ADMoM), distributed training algorithms. Such an approach is proved to converge to the centralized SVM through consensus message exchanges among neighboring nodes.

4.3 Collaborative Decision Tree

Decision tree is another popular supervised learning method, which consists of nodes, branches and leaves. The decision tree is learnt 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 leave node represents the judgement process based features of the data. Information Gain and Gain Ratio can be used to select a feature when generating a decision tree. The basic idea is to reduce $impurity$ in a node. In a distributed scenario, PLANET [40] uses $MapReduce$ to distribute and scale tree induction to 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 [40] [41] [42] adopt horizontal partitioning, i.e., they partition the data. In contrast, YGGDRASILl [43] adopts vertical partitioning based on the features, i.e., each worker stores feature values for even number of features, as well as the labels for the instances for training.

With the development of federated learning, decision tree also is redesigned for the federated platform [44] [45] [46]. Reference [44] proposes 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 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 best features and notifies all the clients. In [45], the authors propose FEDXGB for XGBoost (Extrme gradient boosting) 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 generate CART (Classification and Regression Tree) by finding the optimal split feature. Reference [46] also implements Gradient Boosting Decision Tress (GBDTs) in federated learning framework.

4.4 Collaborative Neural Network (NN)

Since 1980s, neural networks have attracted a lot of attention [47] and recent works [3], [4] have considered collaborative neural networks extensively. Here we mainly focus on the works based on federated learning framework as follows.

4.4.1 Learning Schemes of Federated Learning

Federated averaging is proposed and as a popular model for model aggregation in collaborative NN, which can be expressed as follows:

$$\theta^{(i+1)} = \theta^{(i)} + \frac{1}{n}\sum_{j=1}^{n}\theta_j^{(i)} \tag{8}$$

where $\theta^{(i)}$ is the parameters used in the i^{th} round, n is the number of workers in the i^{th} round, $\theta_j^{(i)}$ represents the parameters updated by the j^{th} user in the i^{th} round. After aggregation, the server distributes $\theta^{(i+1)}$ to all workers for the next round training. With the continuous training and aggregation process, the procedure will stop when certain global accuracy is achieved. To further reduce communication burden between a client and the aggregation server, *structured updates* and *sketched updates* are proposed in [48] to reduce the number of variables and compress them before sending them out. *Structured update* modifies gradients in a restricted space; they design two policies for it which are enforce the update to be a low rank matrix and restrict the update by a random sparse matrix. *Sketched update* first computes full gradient matrix during local training and then compresses it before sending to the root node. FSVRG (Federated Stochastic Variance Reduced Gradient) was proposed in [49] 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 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 [7]. 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 [53] [52] [54] [55], Non-IID data [50] [56] [57] [58].

Although federated learning is good for privacy-preserving services, privacy leakage is possible during the learning procedure, which we address in Section 6.

4.4.2 Resource Constrained Learning

Since federated learning often works in resource-constrained devices under uncertain network environment,

system optimization becomes a critical research topic. There are mainly two approaches to address this.

- **System Optimization:** Here the usage of communication and computation resources is integrated with the federated learning procedure so as to balance prediction accuracy against resource usage. In [59], optimization problem FEDL was proposed by formulating federated learning over wireless network as an optimization problem, and then solved by problem decomposition. Reference [60] proposed to optimize wireless resources and ML based on device selection and receiver beamforming design. The research in [61] further adaptively controls the communication between clients and server to balance the learning accuracy and communication, and the research in [62] optimizes computation offloading in federated learning systems. Also federated learning is adopted in mobile edge computing environment for optimal computing, caching and communication [63]. System optimization also is considered in Geo-Distributed ML in [64].
- **Gradient Quantization:** Gradient quantization uses low-precision values to reduce the communication bandwidth in federated learning. In [65], they proposed 1-bit stochastic gradient decent for speech application which can achieve 10x speedup. In [66] random quantization was used to balance the accuracy and gradient compression. TernGrad [67] further use three-level gradients for quantization.
- **Gradient Sparsification:** Reference [68] randomly drops coordinates of the stochastic gradient vectors and amplifies the remaining coordinates appropriately to ensure the sparsified gradient. Reference [69] shows that almost 99.9% of the gradient exchange in distributed SGD are redundant. Based on this, it proposes Deep Gradient Compression (DGC) to greatly reduce the communication bandwidth. [70] uses Layer Normalization to reduce communication in distributed deep learning.

4.4.3 Coordination Across Workers

In collaborative ML, the Root node may integrate the models trained in multiple ways. Bulk Synchronous Parallelism (BSP), as a classical parallel algorithm, has been widely used in collaborative ML [82]. In BSP, the Root node aggregates parameters from worker nodes, generates a global model, and then sends it to all workers every iteration. However, the if the worker nodes take largely different amounts of time return the response or the communication delays are highly variable, such synchronized approach may be inappropriate. Asynchronous Parallelism (ASP) is the opposite approach where every worker sends its local training parameters to the root node at each iteration without waiting for others. Thus slower workers may send stale parameters to the Root node, and therefore ASP cannot guarantee the model convergence [83]. Stale Synchronous Parallelism (SSP) is a balanced solution between BSP and ASP. SSP restricts the number of iterations between the fastest worker and the slowest worker under a threshold setting in advance [84]. It can reduce the influence of slower workers effectively but the threshold is difficult to set. In order to improve the performance of SSP, Zhao et al. [85] proposed a dynamic SSP (DSSP) method. They use a range of staleness thresholds to determine the number of iterations dynamically. The experiment shows that DSSP are more stable and converges faster than SSP. Shi et al. [86] propose a free SSP (FSSP) strategy to reduce the influence of slow workers. This strategy distinguishes the stragglers which have low efficiency in the entire training process through a penalty mechanism. Persistent stragglers are excluded from the subsequent process to improve the efficiency. The experiment shows that FSSP is 1.5-12 times faster than BSP and SSP.

A summary of discussions in Section 3 and 4 is shown in Table 2.

5 Robustness of collaborative ML

Robustness of collaborative ML (ML) has been discussed widely in recent literature, mostly in the context of federated learning [8], [9], [10], [11].

In general, a ML submodel has four important attributes that may or may not be shared (in addition to the basic API for using the submodel, which we assume is always shared). These are shown in Fig. 7 and explained in the following:

- *Submodel Capabilities:* 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 HA (possibly because HA owns the model), or shared between HA and RA. In the latter case, these could be either provided by the RA to HA as requirements for a private model built by HA, or voluntarily shared by HA with RA.
- *Submodel Structure:* Specifies low level details of the submodel. For example, in case of a decision tree, the structure consists of the entire tree, and in case of a CNN, all of the layers and weights are part of the structure. The structure could either be private to the HA, made visible to RA by HA, or modifiable by RA. In the second case, HA simply builds and trains the model, and then submits it to RA for validation. In the third case, the RA is able to modify the received submodel 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 HA's computing infrastructure.
- *Submodel Training data:* The ability of a HA 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 HA can be valuable even if there is no prohibition on sharing the training data.
- *Submodel Test data:* In all cases of distributed ML, we assume that the RA will not accept a submodel (or results thereof) unless it can pass a specified set of tests on a challenge dataset provided by the RA. If the submodel structure is shared by HA (i.e., a trained submodel delivered by HA 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 HA and expects the test results

TABLE 2
Researches on Collaborative Learning

Types of ML	Collaborative Learning		Collaborative Model	Remarks
Clustering	**Density-based**	DBDC [19] [71]	(b)	Local Specific Core-points
		PartDBSCAN [20]	(b)	R-Tree/Core points
		Mr-DBSCAN [22]	(b)	Core Points/Map-Reduce
		Grid-Based [17]	(a)	Local Specific Core-points
		Boundary-based [26] [27]	(b)	Boundary-based Aggregation
		DENCAST [72]	(b)	Multi-target regression
	Partition-based	PK-Means [73]	(b)	Message-passing based clustering
		Privacy-preserving PK-Means [74]	/	Privacy-preserving
Linear Model	Secure Linear Regression [28]		(b)	Semi-trusted third party
	Privacy-preserving Linear Regression [29]		(b)	Secure multi-party computation
	Secure Logistic Regression [30]		(b)	Homomorphic encryption
SVM	**Parallel-based**	Cascade SVM [31]	(a)	Binary Cascade architecture
		DC-SVM [33]	(b)	Divde and Conquer $\mathcal{O}(n^2/k)$
		PSVM [34]	(b)	$\mathcal{O}(np^2/m)$
		QRSVM [35] [36]	(b)	QR decomposition
	Distribution-based:	DFP-SVM [37]	(a)	Incremental based Aggregation
		Gossip-based [38]	(c)	Fully-dsitributed
		ADMoM [39]	(c)	Fully-dsitributed
Collaborative Decision Tree	**Distributed**	Privacy-DDT [75]	(b)	Homomorphic encryption
		Distributed decision tree v. 2.0 [41]	(a)	Build a DT with Multiple Splits
		YGGDRASIL [43]	/	Deal with compressed data
		PLANET [40]	(b)	Map-reduce
		DT for Robust Regression [42]	(b)	Robustness
		Inprivate digging [76]	(a)	Differential privacy
	Federated	Federated Boosting [75]	(b)	Relaxed privacy constraints
		Federated Forest [44]	(b)	Proposed algorithms at client and master sites
		Federated Gradient Boosting [46]	(b)	Security
Collaborative Neural Network	**Model Aggregation**	FedAvg [77]	(b)	Averaging
		F-Updates [48]		Structure/Sketched updates
		FSVRG [49]		Adaptive to local data
		Federated Meta-Learning [78]		Reduce Communication Cost and Fast Convergence
		Federated matched averaging [79]		Matching and averaging hidden elements
		Agnostic federated Learning [80]		Not biased towards different clients
		Agnostic meta Learning [81]		Global model to be personalized for one client
	System Optimization	FEDL [59]		Wireless Optimization
		In-Edge AI [63]		Optimization based on FL
		FL-Wireless [60]		Wireless
		FL-Control [61]		Adaptive control of communication
		FL-Offloading [62]		Computation Offloading Optimization
		Geo-FL [64]		Geo-Distributed ML
	Gradient Quantization	1-bit SGD [65]		Speech recognition
		QSGD [66]		Random quantization
		TernGrad [67]		Three-level gradients
	Gradient Sparsification	GS [68]		Random Drop-out
		DGS [69]		99% of distribued SGD is redundant
		SDGD [70]		Layer Normalization
	Synchronization	BSP [82]		Bulk Synchronous Parallelism
		ASP [83]		Asynchronous Parallelism
		SSP [84]		Stale Synchronous Parallelism
		DSSP [85]		Determine iteration dynamically
		FSSP [86]		Reduce the influence from slow workers

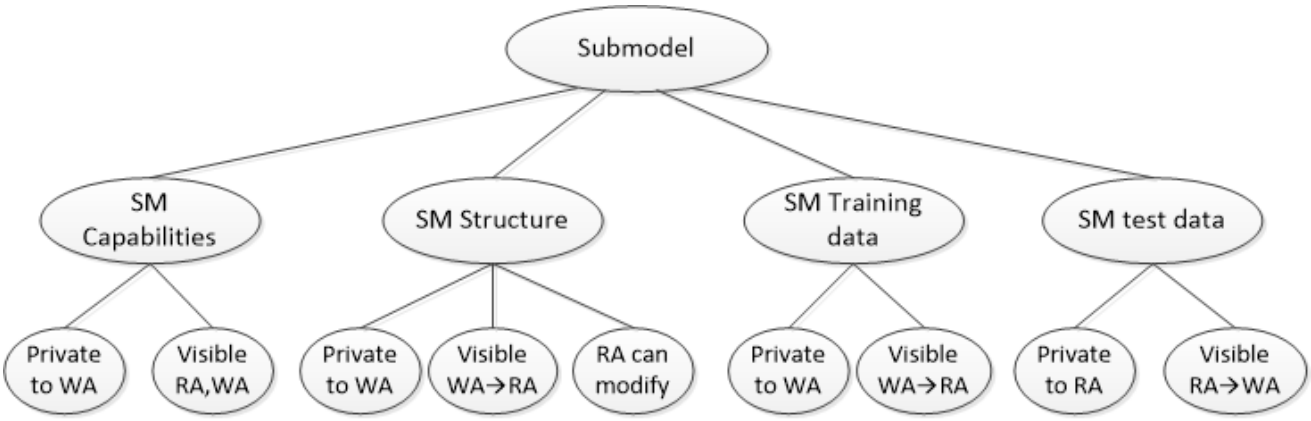


Fig. 7. Illustration of Submodel parameters

back. (According to our assumption, the HA 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.

5.1 SubModel Information Exchange Modes

The model in Fig. 4 admits different scenarios in terms of information exchange between the RA and WAs and the corresponding vulnerabilities. The vulnerability depends on severals aspects including (a) the extent of data/model sharing among RA and WAs and the attacks enabled by this sharing, (b) the attacker capabilities (i.e., level of maliciousness assumed for WAs), (c) the possibility of collusion among WAs, and (d) 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 7. Here, we assume that if HA and RA explicitly stated to share some information or specification, that sharing is considered as a "contract" and remains uncompromised.

Fig. 7 points to many specific scenarios, and the corresponding vulnerabilities. Consider, for example, the case where all 4 items are shared between RA and HA. This corresponds to the situation where the RA provides the requirements to the HA, according to which the HA builds, trains, tests, and then delivers the final submodel 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 HA can take advantage of this knowledge and train the submodel on some additional data that only affects 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 submodel capabilities or the test data with HA, a bad HA does not know how to train the model on bad data without the compromise being revealed. By the same token, a good HA will also have difficulty in meeting the expectations of the RA. In contrast, less sharing by HA (with RA) provides greater opportunities for bad HA to perturb the model or the results. For example, if an HA does not share its training data with RA, a bad HA 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 to the model. In particular, colluding HA's can stagger their spurious output submission in a way to minimize detection. For example, in the case of the same submodel trained by different WAs, a set of bad HA's could supply trained weights to the RA in succession in a way that no improvement takes place over successive training iterations. With identical submodels, 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.

5.2 SubModel Information Exchange vs. Attack Modes

Fig. 7 allows for 16 different scenarios of sharing, each of which can be exploited by a bad HA to carry out various forms of perturbation either during training time or runtime. Most of the attack modes are known in the literature under different names, as discussed below. Reference [87] presents a taxonomy of these modes and comprehensively discusses the papers focusing on these modes. These basic modes can be related to the 16 different possibilities in Fig. 7 in terms of level of visibility (blackbox, greybox and whitebox 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

which can be exploited, i.e., 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 may 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 makes 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 comes 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), so as to force misclassification. Model poisoning happens if training involves iteration; i.e., the RA collects models or model outputs from multiple HA's and uses them to provide feedback to the WAs. For example, if the feedback that depends on the output of one bad HA goes to all the WAs for the next iteration, the entire model gets corrupted or poisoned. Reference [8] discusses 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 an HA is able to deduce the submodel belonging to another HA even though there is no direct revelation of other HA's submodels. The source of this attack is the feedback by RA that includes feedback based on other submodels.

With respect to Fig. 7, it is clear that various attacks are enabled by two aspects of the system: (a) Sharing of information by RA with bad WAs, and (b) Ability of a bad HA to do things beyond the contract (e.g., training with extraneous data).

5.3 Data Poisoning Attacks

Poisoning attacks were first proposed for faking the biometric recognition system. In [88], [89] the authors 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 $x_c = \frac{\sum_{k=1}^{n} x_{ck}}{n}$. When a user submits a sample x, a matching score $s(x, x_c)$ is computed as:

$$s(x, x_c) = 1/(1+||x-x_c||) \tag{9}$$

where $||.||$ 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: 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 of biometric patterns, such as aging. The self-update is accepted if $s(x, x_c) > \theta_c$, where θ_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. The authors in [88] have studied that in the infinite window scenario, the number of attack samples must grow exponentially with $||x_a - x_c||$, whereas in case of finite window the number of samples must grow linearly. Thus the latter is more vulnerable to poisoning attack as compared to the former scenario. Poisoning attacks are also studied for anomaly detection in [90], [91].

In [92] the authors have shown that by applying an imperceptible non-random perturbation to a test image (which are termed as *adversarial examples*), it is possible to fool a DNN to arbitrarily change the network's prediction. In [93] the authors have argued that the primary cause for such vulnerability to adversarial perturbation is their linear nature. Similar poisoning attacks on support vector machines are studied in [94], [95]. In [96] the authors have shown that by injecting around 50 dirty-label samples, a backdoor adversary can achieve an attack success rate of above 90%.

In [97] the authors have discussed data poisoning attacks on neural networks using direct gradient methods. They have also proposed a generative method to speed up the generation of poisoning data by using the inspiration from Generative Adversarial Network (GAN), and have shown that the generative method speeds up the poisoned data generation by 239.38 times as compared to the direct gradient method. In [98] the authors have proposed a *back-gradient optimization* to launch poisoning attacks on neural networks and deep learning architectures, which is more computationally efficient than gradient-based poisoning attacks. In [99] the authors have studied poisoning attacks on federated learning system based on GAN, where an attacker can stealthily trains a GAN to mimic the prototypical samples of the other workers. The GAN can then be controlled for generating the poisoning updates.

5.4 Model Poisoning Attacks

Several types of model poisioning attacks are possible in Fig. 4. 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 HA 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 then a certain threshold a^*, i.e.

$$\mathcal{A}(F_{\Theta'}, D_{\text{valid}}) \geq a^* \tag{10}$$

In this case, the HA can return a maliciously backdoored model such that, (a) the returned model does not reduce the classification accuracy of the validation set, and (b) for certain inputs containing the backdoor trigger, the prediction of the outputs are 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 HA "fix" the problem, it provides a mechanism for the malicious HA to check what kind of backdoor to not use.

In [10] the authors 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 mis-classification with a success rate of 99%. In [100] the authors have developed a model poisoning attack where the malicious worker can use model replacement to introduce backdoor functionality. The authors have shown that such model replacement attack greatly outperforms the conventional data poisoning attack.

Reference [11] shows that the success rate of model poisoning increases linearly with the number of poisoned samples and number of attackers. The paper also proposes a filtering defense based on the distance between model updates by honest and malicious agents. Reference [101] discusses a variant of model poisoning attack, called data contamination attack, that 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. The paper shows that adversarial training can mitigate such attacks. Another variant is a form of sybil attack discussed in [102] where the a malicious agent mimics the actions of a legitimate agent but alters the training data to force misclassification.

5.5 Blackbox, Greybox and Whitebox Attacks

Adversarial ML is often broadly categorized into black-box, grey-box and white-box attacks [103], 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 the extent of knowledge is quantified, as is done in our model in Fig. 7. Note that anything that a HA can keep private could potentially be altered by it, possibly maliciously. Even in 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 HA agree to use certain data for training, it is indeed used), since otherwise a classification is not very meaningful.

In [104] the authors have studied black-box 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 good attacker loss function for each defense. In [105] the authors have discussed the interaction between an adversary and deep learning model as a two-player sequential non-cooperative 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 a correct classification and the adversary creates new instances of data using genetic operations for mislead the classification. The game is solved by the Nash equilibrium, which leads to solutions that are robust to subsequent adversarial data manipulations. In [106] the authors 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 have shown that the DNN misclassifies 84.24% of the adversarial inputs. Reference [107] has developed generalized black-box attacks by exploiting adversarial sample transferability on a broad classes of ML classifiers, including neural networks, logistic regression, support vector machines, decision trees, nearest neighbors, and ensembles. In [108] the authors 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-box and black-box attacks.

Reference [109] has 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.

In [110] the authors have used the concept of "defensive distillation" as a defensive mechanism against adversarial samples, by reducing the amplitude of neural network 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, in [111] the authors have created a set of white-box attacks that successfully find adversarial examples for 100% of images on defensively distilled networks. Reference [112] has discussed a class of algorithms for adversarial sample creation against any feed-forward 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 [113], [114], [115], [116]. Adversarial robustness of neural networks against both black-box and white-box attacks are discussed in [117]. A summary of representative attackers are discussed in Table 3.

TABLE 3
A 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 [88]	Perfect knowledge	Template self-update/face verification
		Reference [89]	Limited knowledge	Face verification
		Reference [90]	Zero knowledge	Classification
		Reference [91]	Full/limited control over training data	Anomaly detection
		Reference [92]	Limited knowledge	Image classification
		Reference [94]	Perfect knowledge	Image classification
		Adversarial label flips attack [95]	Perfect knowledge	Binary classification
		Backdoor attacks [96]	Zero knowledge	Image classification
		Generative Poisoning Attack [97]	Perfect knowledge	Image classification
		Back-gradient optimization [98]	Perfect/limited knowledge	Spam filtering, malware detection, handwritten digit recognition
		Generative Adversarial Nets [99]	Perfect/limited knowledge	Image classification
		Reference [104]	Perfect/zero knowledge	Image classification
Model Poisoning attacks	Attacker changes the model output to force misclassification	Reference [10]	Limited knowledge	Image/traffic sign classification
		Reference [100]	Limited knowledge	Image classification
		Reference [101]	Limited knowledge	Multi-party environment
		Reference [102]	Limited knowledge	Image classification
Black box attacks	Attacker has zero knowledge about the ground truth sand the learning mechanism	Reference [104]	Zero/limited knowledge	Image classification
		Reference [105]	Zero knowledge	Image classification
		Reference [106]	Zero knowledge	Image classification
		Reference [107]	No knowledge	Image classification
		Momentum iterative gradient-based methods [108]	No knowledge	Image classification
White box attacks	Attacker has full knowledge about the learning mechanism and the ground truth	Reference [104]	Perfect knowledge	Image classification
		Robust Physical Perturbations [109]	Perfect knowledge	Road sign/image classification
		Reference [111]	Perfect knowledge	Image classification
		Reference [112]	Perfect knowledge	Image classification
		Reference [114]	Limited/Perfect knowledge	Malware detection
		Reference [115]	Perfect knowledge	Image classification
		Reference [116]	Perfect knowledge	Image classification

6 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 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 the discuss current solutions trying to solve them.

6.1 Membership Inference

The goal of membership inference is to infer if certain data set was used in the training. Take document survey as an example, the adversary who plans a membership inference attack, wishes to confirm whether an individual participated in the survey [118].

In [119], Shorti et. al. proposed a framework for membership inference in a centralized learning scenario, and it worked extremely well even in the public ML service like Google and Amazon. For the convolutional neural network which uses softmax function in the final layer for classification, the value in each dimension from the softmax function can represent the probability that the input belongs

to that category. The training continuously reduces the gap between the hypothesized function and the obtained results, which makes the probability for privacy leakage higher and higher. This phenomenon is called *model confidence*, and they define *higher confidence* as the higher probability bias in the right dimension compared to the other wrong dimension. 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 thus the *model confidence* can be a possible reason for membership inference.

Therefore, when an attacker has samples similar to that of the victim user, they can use those samples to construct a similar model with the same training and network structure. The author uses three kinds of methods to construct the victim users's fake dataset, and the membership inference attack shows excellent performance with different dataset and network structure.

However, the reason of the successful attack is still not so clear. Many researchers think that it happens because the neural network is *over-fitted*. In fact, with the *early stop* and *regularization*, the attack in the previous work [119] performs worse, but still works better than the baseline of *random guess*. So the extension of this topic can be divided into two approaches: one is to figure out the relationship between over-fitting and membership inference attack, and the other is to extend the membership inference in different scenarios. For the first approach, Song et. al. investigated the over-fitting scenario by simulating a malicious ML provider, who provides ML algorithms for other users as services [120]. They evaluated memorizing users' data by a ML algorithm in both white-box 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 the model finally over-fitted the synthetic data, and could be one reason for privacy-leakage. Leino, Yeom et. al. have also investigated this problem in [121] [122]. Authors in [121] evaluated over-fitting and privacy-leakage quantitatively by targeting on membership inference and attribute inference. They found over-fitting 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 over-fit) while attribute inference is more sensitive to over-fitting.

For the second approach, Long et. al. analyzed the association between generalization and membership inference in [118], and found that the participation of different sample data results in different models. Thus the success of membership inference may be caused by some unique influence due to the participation of different samples. In 2019, Nasr, et.al. [123] proposed a new framework for membership inference attack based on federated learning, considering over-fitting. They pointed out that in the later stages of federated learning, there's a phenomenon that the testing samples' gradients are always greater than that of the training samples. Therefore, they extend the input of classifier in [119], take the gradients in each layer of neural network into consideration, and shows the accuracy of the attack can reach to 80%.

The membership inference in federated learning scenario is still a new area of exploration, and generalized attack scenarios and their systematic analysis are still lacking.

6.2 Unintended Feature Inference

In an Unintended Feature Inference attack, the attacker's goal is to deduce features that are irrelevant for the the trained prediction model. For example, a convolutional neural network to do face recognition can grasp some unintended features of the dataset such as whether a specific person in the training dataset wears 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.

Song et al. proposed this interesting attack in collaborative learning and federated learning scenario [124]. 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 he wants to infer, whereas the other one does not. Collecting certain amount 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 in federated learning.

6.3 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 main representative information of the training sample, called representative sample.

Representative sample reconstruction attack is generally performed through two approaches: one by applying Generative Adversarial Nets (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 user's data can expose a lot of information. A representative sample consisting of characteristics of workers' data can be used to perform poisoning attack [9], which could also substantially harm the global accuracy of federated learning.

In 2017, Hitaj et al. proposed the first representative sample reconstruction framework in federated learning [125]. This attack needs to be performed in a classifier model training scenario, such as CNN. For example, consider a CNN for the recognition of hand-written digits. The traditional model here has an output vector in 10 dimensions. An attacker can add a new dimension into the output vector and thereby completely control the structure of federated learning. This reconstructed model will be seen as the global training model. The attacker then creates two models with the same structure at the same time, one is called *global model* for parameters update, and the other is called *local model* for attacking. During the new round for parameter

update in the federated learning, the attacker will download the parameters updated by the victim-user to initialize his *local model*, and download the parameters from server to initialize his *global model*, after that, he will use the *local model* as *Discriminator* of GAN, and update his *Generator* with the following equation:

$$\min_{\theta_G} \sum_{i=1}^{n} \log(1-f(G(x_i;\theta_G);\theta_D)) \tag{11}$$

where θ_G and θ_D stand for the parameters of *Generator* and the *Discriminator* respectively, functions G and f denote the outputs of the *Generator* (the fake sample) and *Discriminator* in the dimension corresponding to the input's label. Finally n represents the number of samples that the *Generator* generated.

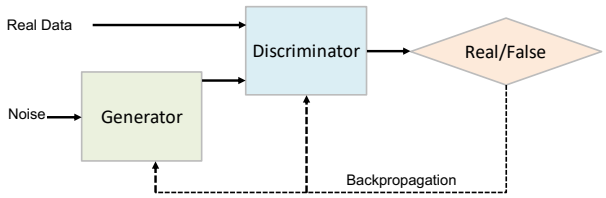


Fig. 8. Illustration of GANs

Fig. 8 shows the generic way that GANs operate. It involves two models, one is *Discriminator* and the other *Generator*. *Discriminator* is based on discriminative model, which is trained based on sample data (real data). While *Generator* follows generative model, which generates observation data based on some hidden information as input data. In GAN, *Generator* generates a sample data based on a given noise, and then *Discriminator* judges whether the input data is real or false. Two models are trained iteratively through back-propagation. The attacker uses the generated sample data and changes its label to the final class (i.e. the 11_{th} class), adds it to the training set of the global model, and finally trains a model that can recognize 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 of 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 Model Inversion [126], since both of them aim to recover the training data by extracting the characteristic information. However, they are slightly different in the following two aspects. First, Model Inversion is used to attack a centralized learning model, while the Representative Sample Reconstruction is used in collaborative learning [125], [127]. Second, Model Inversion is based on the *confidence* of model, but actually, some researches have found that the trained model still can be fooled by randomly generating some strange sample. For example, the trained model can recognize some Gaussian noise as digital numbers, which leads to classification error, i.e., recognizing a sample data which does not match with the human-being's perspective [128] [129]. So, Model Inversion can obtain sensitive information, but also probably get the meaningless information [125]. To avoid this problem, Representative Sample Reconstruction attack improves it with Generative Adversarial Networks(GAN) [130], which not only ascertains the extraction of characteristic, 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. [125], Wang et al. further extended this type of attack which is called *mGAN-AI* [127]. Based on the idea that all users train the *Discriminator*, this GAN adds another procedure as follows. The algorithm also catches the parameters updated by the victim-user to update the *Discriminator* as in the previous work. However, *mGAN-AI* also adds an additional step to generate some sample data through parameters by using L-BFGS algorithm [131], 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 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.

6.4 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* [132] [133], *secure multiparty computation* [134] [135] [136], and *homomorphic encryption* [137], [138]. In this subsection, we discuss these frameworks in detail.

Informally, the goal of the differential privacy is to maximize the availability of data with limited sensitive information leakage. In 2006, Netflix hosted a competition to improve its algorithm for providing movie recommendations to customers based on their past choices. To protect privacy, they hid users' id associated with the recommendations. But by searching the keyword of each recommendation, it was still easy to find the the corresponding user of each comment. To solve this kind of problem, C. DWork et. al. proposed a privacy preserving approach, called differential privacy [132]. The main idea of this method is to add noise into the information for privacy guarantee from a mathematical perspective. This approach has been used in many scenarios, and plays an important role in the privacy preserving of federated learning. In 2015, Shorki and Shmatikov proposed a privacy preservation framework in collaborative learning [139]. Compared to the original federated learning, in this framework each worker selects a part of parameters and adds noise by differential privacy before uploading. This gives a much more promising privacy guarantee for selection upload with noise [119] [124]. However the latter research also pointed out, the framework may lead to an extremely slow global convergence in a small scale federated learning system [124]. Also it may not work well in avoiding *representative sample reconstruction* attack [125].

As the later work pointed out, leakage of privacy still exists even with Differential Privacy. Since the model itself can represent the characteristics of the user, there is a privacy leakage when the model needs to be shared among many workers during for federated learning. To avoid this type of privacy leakage with good usage of all the workers'

data, researchers proposed a combined way to integrate *Homomorphic Encryption* and *Secure Multi-Party Computation*. *Homomorphic Encryption Algorithm* 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:

$$Enc(x_1+x_2) = Enc(x_1) \circ Enc(x_2) \tag{12}$$

where the function *Enc(.)* represents an encryption to translate a plain text into a cipher text, x_1 and x_2 are two variables, and $\circ$ represent a operator for cipher text. This algorithm provides a framework of *secure multi-party computation*, enable workers have a stronger privacy guarantee.

In 2017, Phong et al. tried to apply homomorphic encryption algorithms to address privacy preservation in federated learning [140]. In this work, all the workers apply homomorphic encryption and update the encrypted parameters to the server. After combination in the server, they download the encrypted parameters, and decrypt them to start a new round of federated learning. This framework works well in avoiding the attacks from *curious-but-honest* server (i.e. the server will obey the federated learning protocol honestly, but very curious about the data of the workers). However, the framework still has some open problems. For example, this framework can't deal with the attack scenario where the workers and server conspire. To address it, we need to introduce a trusted third party [141], or improve the algorithm with multi-key homomorphic encryption algorithm proposed [142]. On the other hand, because the gradient is not visible to the server, it is difficult to analyze the appropriateness of each user's parameter set. This would allow a poisoning attack possibly leading to significant impact on global accuracy [9].

Table 4 summarises the privacy attack technology and privacy-preserving framework.It consists of representative works for the three types of privacy attack, and possible ways to protect privacy by considering a honest-but-curious server and an active adversary respectively.

7 Discussion and Challenging Issues

7.1 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 submodel decomposition and integration are still not investigated well. For example, while clustering, linear model, and SVM all benefit from Incremental Model Integration, they need to be further investigated in federated learning. Also, the scalability of distributed optimization methods in federated learning still needs more studied. We find that most of collaborative learning schemes follow passive and fixed participating model, which means 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, etc. For example, active participation requires new coordination mechanisms and their convergence analysis, which may be quite challenging.

Meanwhile, for optimization issue in federated learning, there still needs to investigate learning model in theoretically by jointly considering wireless channel, mobility model of users etc., to expand federated learning in future applications, such as training an autonomous driving model by aggregating a lot of real drivers. Meanwhile, iteration time slot and frequency should be joint optimized accordingly for a mobile user scenario. Also considering mobile user case, synchronization schemes shall further be investigated to take care of data loss, synchronization error, communication error, and other cases, by guaranteeing learning model works well in such scenarios and can converge to a accurate model. Finally, parameter compression also should be further studied to be more efficient, and is also better to be considered joint with privacy issue.

7.2 Robustness

The general models for distributed ML shown in Figs. 4 and 7 point to a large number of scenarios, many of which can be very challenging to address. The most studied special case so far is the federated learning where a common shared model is trained by all WAs using private data and then the model is run by RA. As discussed earlier, many attack modes are possible in this case including data poisoning, model poisoning, backdoor, evasion, sybil attacks, etc. Mitigation of many of these perturbations remains challenging, in spite of several methods discussed in the literature. 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 ML, this amounts to ensuring that the model does not provide an anomalous answer because of illicit training on 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 the 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 overall model consisting of multiple submodels, each submodel needs to be made robust. This would require, at a minimum, that the RA validate each submodel separately before fusing their 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, mechanism is to hand out each submodel for training to multiple WAs, exactly as in federated learning. For more sophisticated approaches, the decomposition of the model into submodels needs to be such that each feature is covered by multiple submodels. This can be challenging since the submodel 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 submodel outputs to determine vulnerabilities.

TABLE 4
A Summary of Privacy Attacks and Protection Frameworks

Types of the Privacy Attack	Attack Methods	Representative Works		Several Possible Way to Protect	
Membership Inference	Given a trained model and a data, the attacker wants to figure out whether this sample belongs to the training set or not.	Taking advantage of the *model confidence* to train a classifier, and judge whether the target model has a clear bias on that dimension in softmax layer [119]		Honest-But-Curious	Updating part of parameters with noise to guarantee privacy [139]
		Training Several Models with the same structure and data extract from same distribution to build CDF function [118]			Using *Homomorphic Encryption* to build the *Secure Multi-Party Computation* [140] [143]
		Making a comprehensive analytic of the *Model Confidence* and the *Gradient* [123]		Active Adversary	Referring to the *HTTPs* communication algorithm to make the updated parameters invisible to the *third party*, and add the special noise to it [141]
Unintended Feature Inference	Given a pre-trained model, figure out some features not directly relevant to the training	Training two kinds of model, one is trained with the dataset having the feature the attacker wants to infer, while the model's dataset do not have [124]		The Experiment shows that this attack is restrained to many condition.	
Representative Sample Reconstruction	Attack a certain federated learning worker by extracting the characteristics in the training set, and create a sample that can not distinguish with the training data	Model Inversion	Using Heuristic Algorithm, iteratively construct a sample that has the minimize loss w.r.t the given pre-trained model. [126]	Honest-But-Curious	Using *Homomorphic Encryption* to build the *Secure Multi-Party Computation* [140] [143]
		Representative Sample Reconstruction	The attack comes from a federated worker. After generating the representative sample with *Generator*, attacker will give the synthetic sample a wrong label, and put it into the training set [125]	Active Adversary	Referring to the *HTTPs* communication algorithm to make the updated parameters invisible to the *third party*, and add the special noise to it, which is secretly created by the workers, and can be offset by other updated parameters. [141]
			The attack is coming from the aggregation server. Attacker use *L-BFGS* algorithm to compute each user's identity sample, and then train the *Discriminator* [127]		

7.3 Privacy

We discussed the main privacy attacks including Membership Inference, Unintended Feature Inference and Representative Sample Reconstruction, and the corresponding privacy-preserving frameworks. The current representative works on the latter are mainly based on *secure multiparty computation*, which can provide a secure environment for parameter aggregation. In such an environment, one user cannot read the parameters of other users. However, if there is a malicious user who intends to upload bad parameters to the root server, it is possible for him to avoid the detection from other users, and this ultimately affects the global model. The representative work is poisoning attack as described in Section 5. For example, attackers can generate representative samples by GAN and give them a wrong label intentionally. Then the model can predict wrong output with some specific features [100] [9]. The attack result can be serious, e.g., the agent can treat a stop sign as a speed limit sign to make a wrong decision in intelligent vehicles.

Malicious user/parameter detection becomes even more important under privacy-preserving framework since the parameters are non-transparent to other collaborative nodes. Also, the computation and communication costs are rather high in the current privacy-preserving frameworks,

e.g., *secure multiparty computation* and *homomorphic encryption algorithm*. How to balance the training prediction and privacy-preserving is still an open issue.

8 Conclusion

In this paper, 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 the privacy, the existing work shows that over-fitting 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 are crucial for their widespread use.

Acknowledgments

The work was partially supported by the faculty starting funding from the Sun Yat-sen University. Parts of the works are extension from the work under JST-NSF Joint Funding to study Big Data and Disaster (SICORP Project). We would like to thanks Mr. Feiyuan Liang who is a master student in Sun Yat-sen University for help with literature review.

References

[1] D. Peteiro-Barral and B. Guijarro-Berdiñas, "A survey of methods for distributed machine learning," *Progress in Artificial Intelligence*, vol. 2, no. 1, pp. 1–11, 2013.

[2] S. Devi, "A survey on distributed data mining and its trends," *International Journal of Research in Engineering & Technology (IJRET)*, vol. 2, no. 3, pp. 107–120, 2014.

[3] R. Gu, S. Yang, and F. Wu, "Distributed machine learning on mobile devices: A survey," *arXiv preprint arXiv:1909.08329*, 2019.

[4] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–43, 2019.

[5] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *arXiv preprint arXiv:1912.09789*, 2019.

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

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[8] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," *arXiv preprint arXiv:1811.12470*, 2018.

[9] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2019, pp. 374–380.

[10] S. Shen, S. Tople, and P. Saxena, "Auror: defending against poisoning attacks in collaborative deep learning systems," in *ACSAC*, 2016, pp. 508–519.

[11] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2019, pp. 233–239.

[12] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[13] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, 2019.

[14] N. Kumar, S. Misra, and M. S. Obaidat, "Collaborative learning automata-based routing for rescue operations in dense urban regions using vehicular sensor networks," *IEEE Systems Journal*, vol. 9, no. 3, pp. 1081–1090, 2014.

[15] W. Tang, K. Zhang, D. Zhang, J. Ren, Y. Zhang, and X. S. Shen, "Fog-enabled smart health: Toward cooperative and secure healthcare service provision," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 42–48, 2019.

[16] P. Verma and S. K. Sood, "Fog assisted-iot enabled patient health monitoring in smart homes," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1789–1796, 2018.

[17] 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 Transactions on Parallel and Distributed Systems*, 2019.

[18] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[19] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "Dbdc: Density based distributed clustering," in *International Conference on Extending Database Technology*. Springer, 2004, pp. 88–105.

[20] X. Xu, J. Jäger, and H.-P. Kriegel, "A Fast Parallel Clustering Algorithm for Large Spatial Databases," *Data Mining and Knowledge Discovery*, vol. 3, no. 3, p. 27, 1999.

[21] T. Sakai, K. Tamura, K. Misaki, and H. Kitakami, "Parallel processing for density-based spatial clustering algorithm using complex grid partitioning and its performance evaluation," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, 2016, p. 337.

[22] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan, "Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce," in *IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS)*, 2011, pp. 473–480.

[23] C. Böhm, R. Noll, C. Plant, and B. Wackersreuther, "Density-based clustering using graphics processors," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 2009, pp. 661–670.

[24] B. Welton and B. P. Miller, "Data Reduction and Partitioning in an Extreme Scale GPU-Based Clustering Algorithm," *2nd International Workshop on Data Reductions for Big Scientific Data (DRBSD), Denver*, 2017.

[25] B. Welton, E. Samanas, and B. P. Miller, "Mr. scan: Extreme scale density-based clustering using a tree-based network of gpgpu nodes," in *2013 SC - International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2013, pp. 1–11.

[26] M. Bendechache and M. T. Kechadi, "Distributed clustering algorithm for spatial data mining," in *2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM)*, July 2015, pp. 60–65.

[27] J.-F. Laloux, N.-A. Le-Khac, and M.-T. Kechadi, "Efficient distributed approach for density-based clustering," *Proceedings of the 2011 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2011*, no. Cdm, 2011.

[28] F. K. Dankar, R. Brien, C. Adams, and S. Matwin, "Secure multiparty linear regression." in *EDBT/ICDT Workshops*. Citeseer, 2014, pp. 406–414.

[29] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 345–364, 2017.

[30] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: Design and evaluation," *JMIR medical informatics*, vol. 6, no. 2, p. e19, 2018.

[31] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel support vector machines: The cascade svm," in *Advances in neural information processing systems*, 2005, pp. 521–528.

[32] Y. You, J. Demmel, K. Czechowski, L. Song, and R. Vuduc, "Ca-svm: Communication-avoiding support vector machines on distributed systems," in *2015 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2015, pp. 847–859.

[33] C.-J. Hsieh, S. Si, and I. Dhillon, "A divide-and-conquer solver for kernel support vector machines," in *International conference on machine learning*, 2014, pp. 566–574.

[34] 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.

[35] J. Dass, V. Sarin, and R. N. Mahapatra, "Fast and communication-efficient algorithm for distributed support vector machine training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 1065–1076, 2018.

[36] J. Dass, V. P. Sakuru, V. Sarin, and R. N. Mahapatra, "Distributed qr decomposition framework for training support vector machines," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 753–763.

[37] K. Flouri, B. Beferull-Lozano, and P. Tsakalides, "Training a svm-based classifier in distributed sensor networks," in *2006 14th European Signal Processing Conference*. IEEE, 2006, pp. 1–5.

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

[39] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[40] B. Panda, J. S. Herbach, S. Basu, and R. J. Bayardo, "Planet: massively parallel learning of tree ensembles with mapreduce," 2009.

[41] A. Desai and S. Chaudhary, "Distributed decision tree v. 2.0," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 929–934.

[42] T. Guo, K. Kutzkov, M. Ahmed, J.-P. Calbimonte, and K. Aberer, "Efficient distributed decision trees for robust regression," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 79–95.

[43] F. Abuzaid, J. K. Bradley, F. T. Liang, A. Feng, L. Yang, M. Zaharia, and A. S. Talwalkar, "Yggdrasil: An optimized system for training deep decision trees at scale," in *Advances in Neural Information Processing Systems*, 2016, pp. 3817–3825.

[44] Y. Liu, Y. Liu, Z. Liu, J. Zhang, C. Meng, and Y. Zheng, "Federated forest," *arxiv.org/abs/1905.10053*, 2019.

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

[46] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," *arXiv preprint arXiv:1911.04206*, 2019.

[47] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation applied to handwritten zip code recognition, 1989," *Dostupné z: http://yann. lecun. com/exdb/publis/pdf/lecun-89e. pdf*.

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

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

[50] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," 07 2019.

[51] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016.

[52] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, 2019.

[53] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[54] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, pp. 1–1, 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 Transactions on Industrial Informatics*, 2019.

[56] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2019.

[57] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," *arXiv e-prints*, p. arXiv:1806.00582, Jun 2018.

[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," *arXiv e-prints*, p. arXiv:1811.11479, Nov 2018.

[59] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1387–1395.

[60] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning based on over-the-air computation," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[61] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 63–71.

[62] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," *IEEE Access*, vol. 7, pp. 69 194–69 201, 2019.

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

[64] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching {LAN} speeds," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 629–647.

[65] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[66] D. Alistarh, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Randomized quantization for communication-optimal stochastic gradient descent," *arXiv preprint arXiv:1610.02132*, 2016.

[67] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in neural information processing systems*, 2017, pp. 1509–1519.

[68] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 1299–1309.

[69] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[70] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.

[71] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "Scalable density-based distributed clustering," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2004, pp. 231–244.

[72] R. Corizzo, G. Pio, M. Ceci, and D. Malerba, "Dencast: distributed density-based clustering for multi-target regression," *Journal of Big Data*, vol. 6, no. 1, p. 43, 2019.

[73] M. N. Joshi, "Parallel K - Means Algorithm on Distributed Memory Multiprocessors," *Cities*, p. 12, 2003.

[74] S. Merugu and J. Ghosh, "Privacy-preserving distributed clustering using generative models," in *Third IEEE International Conference on Data Mining*. IEEE, 2003, pp. 211–218.

[75] W. Fang, B. Yang, D. Song, and Z. Tang, "A new scheme on privacy-preserving distributed decision-tree mining," in *2009 First International Workshop on Education Technology and Computer Science*, vol. 2. IEEE, 2009, pp. 517–520.

[76] L. Zhao, L. Ni, S. Hu, Y. Chen, P. Zhou, F. Xiao, and L. Wu, "Inprivate digging: Enabling tree-based distributed data mining with differential privacy," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2087–2095.

[77] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

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

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

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

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

[82] J. Jiang, B. Cui, C. Zhang, and L. Yu, "Heterogeneity-aware distributed parameter servers," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 463–478.

[83] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[84] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Advances in neural information processing systems*, 2013, pp. 1223–1231.

[85] X. Zhao, A. An, J. Liu, and B. X. Chen, "Dynamic stale synchronous parallel distributed training for deep learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1507–1517.

[86] H. Shi, Y. Zhao, B. Zhang, K. Yoshigoe, and F. Chang, "Effective parallel computing via a free stale synchronous parallel strategy," *IEEE Access*, vol. 7, pp. 118 764–118 775, 2019.

[87] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Computer Science Review*, vol. 34, p. 100199, 2019.

[88] B. Biggio, G. Fumera, F. Roli, and L. Didaci, "Poisoning adaptive biometric systems," in *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, SSPR&SPR*, 2012, pp. 417–425.

[89] B. Biggio, L. Didaci, G. Fumera, and F. Roli, "Poisoning attacks to compromise face templates," in *International Conference on Biometrics*, 2013, pp. 1–7.

[90] B. Nelson and A. D. Joseph, "Bounding an attack' s complexity for a simple learning model," 2006.

[91] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact," in *AISTATS*, 2010, pp. 405–412.

[92] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[93] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.

[94] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *ICML*, 2012.

[95] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *ECAI*, 2012, pp. 870–875.

[96] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *CoRR*, vol. abs/1712.05526, 2017.

[97] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative poisoning attack method against neural networks," *CoRR*, vol. abs/1703.01340, 2017.

[98] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," *CoRR*, vol. abs/1708.08689, 2017.

[99] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *IEEE TrustCom/BigDataSE*, 2019, pp. 374–380.

[100] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *CoRR*, vol. abs/1807.00459, 2018.

[101] J. Hayes and O. Ohrimenko, "Contamination attacks and mitigation in multi-party machine learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 6604–6615.

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

[103] 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, 2019.

[104] N. Carlini and D. A. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," *CoRR*, vol. abs/1705.07263, 2017.

[105] A. S. Chivukula and W. Liu, "Adversarial learning games with deep learning models," in *IJCNN*, 2017, pp. 2758–2767.

[106] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ACM AsiaCCS*, 2017, pp. 506–519.

[107] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *CoRR*, vol. abs/1605.07277, 2016.

[108] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *IEEE CVPR*, 2018, pp. 9185–9193.

[109] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *IEEE CVPR*, 2018, pp. 1625–1634.

[110] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *CoRR*, vol. abs/1511.04508, 2015.

[111] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016.

[112] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE EuroS&P*, 2016, pp. 372–387.

[113] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *ICLR*, 2017.

[114] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," *CoRR*, vol. abs/1708.06131, 2017.

[115] J. Hayes and G. Danezis, "Learning universal adversarial perturbations with generative models," in *IEEE SP Workshops*, 2018, pp. 43–49.

[116] A. Athalye, N. Carlini, and D. A. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *ICML*, 2018, pp. 274–283.

[117] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[118] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018.

[119] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[120] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *CCS '17*, 2017.

[121] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 268–282.

[122] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," *arXiv preprint arXiv:1906.11798*, 2019.

[123] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks," *arXiv preprint arXiv:1812.00910*, 2018.

[124] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019*

IEEE Symposium on Security and Privacy (SP). IEEE, 2019, pp. 691–706.
[125] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 603–618.
[126] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," p. 1322–1333, 2015. [Online]. Available: https://doi.org/10.1145/2810103.2813677
[127] 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 *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.
[128] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014. [Online]. Available: http://arxiv.org/abs/1312.6199
[129] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," pp. 1–10, 01 2015.
[130] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 2672–2680.
[131] "Wikipedia : Limited-memory bfgs algorithm," https://en.wikipedia.org/wiki/Limited-memory_BFGS.
[132] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *EUROCRYPT*, 2006.
[133] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014. [Online]. Available: http://dx.doi.org/10.1561/0400000042
[134] "Wikipedia : Secure multi-party computation," https://en.wikipedia.org/wiki/Secure_multi-party_computation#cite_note-1.
[135] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, Oct 1986, pp. 162–167.
[136] M. Yung, "From mental poker to core business: Why and how to deploy secure computation protocols?" in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1–2. [Online]. Available: https://doi.org/10.1145/2810103.2812701
[137] "Sciencedirect : Homomorphic encryption," https://www.sciencedirect.com/topics/computer-science/homomorphic-encryption.
[138] P. Parmar, S. Padhar, S. Patel, N. Bhatt, and R. Jhaveri, "Survey of various homomorphic encryption algorithms and schemes," *International Journal of Computer Applications*, vol. 91, 03 2014.
[139] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.
[140] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.
[141] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.
[142] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems*, vol. 74, pp. 76–85, 2017.
[143] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," 11 2019, pp. 1–11.

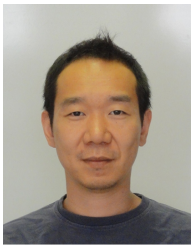
Junbo Wang received his Ph.D. degree from the University of Aizu, Japan in computer science and engineering in 2011. He was a Postdoctoral and Associate Professor in the Univeristy of Aizu, Japan. Current he is an Associate Professor in School of Intelligent Systems Engineering, Sun Yat-sen University, China. He was the PI at Japan site for JST-NSF Joint Funding to study Big Data and Disaster (SICORP Project) cooperating with Dr. Krishna Kang and Dr. Amitangshu Pal. His research interests include collaborative ML, federated learning, fog computing, big and privacy.

Amitangshu Pal received the B.E. degree in computer science and engineering from Jadavpur University, in 2008, and the Ph.D. degree in electrical and computer engineering from The University of North Carolina at Charlotte, in 2013. He was a Postdoctoral Scholar with Temple University, where he is currently an Assistant Professor with the Computer and Information Science Department. He has published over 50 conferences and journal papers. His current research interests include wireless sensor networks, reconfigurable optical networks, smart health-care, cyber-physical systems, mobile and pervasive computing, and cellular networks.

Kaiming Zhu received the B.E. degree in computer science and engineering from Northeastern University, China, in 2019, and now he is a master student with Sun Yat-sen University, China. His research interests include federated learning, privacy-preserving mechanism.

Krishna Kant is currently a professor in the Computer and Information Science Department at Temple University in Philadelphia, PA where he directs the IUCRC center on Intelligent Storage. Earlier he was a research professor in the Center for Secure Information Systems at George Mason University. From 2008-2013 he served as a program director at NSF where he managed the computer systems research program and was instrumental in the development and running of NSF-wide sustainability initiative named science, engineering and education for sustainability (SEES). Prior to NSF, he served in industry for 18 years (at Intel, Bellcore, and Bell Labs) and 10 years in academia (at Penn State and Northwestern Univ.). He carries a combined 40 years of experience in academia, industry, and government. He received his Ph.D. degree in Mathematical Sciences from the University of Texas at Dallas in 1981. He has published in a wide variety of areas in computer science, authored a graduate textbook on performance modeling of computer systems. His research interests span a wide range including energy efficiency, robustness, and security in cyber and cyber-physical systems. He is a Fellow of the IEEE.

Wuhui Chen received his bachelor?s degree in 2008 from Software College Northeast University, China. He received his master?s and doctoral degrees from School of Computer Science and Engineering, University of Aizu, Japan in 2011 and 2014, respectively. Now he is an associate professor at Sun Yat-sen university, China. He was a researcher at the Revitalization Center, University of Aizu, Japan. His research interests include services computing, edge computing, and cloud robotics.



Song Guo (M'02-SM'11) is a Full Professor at Department of Computing, The Hong Kong Polytechnic University. His research interests are mainly in the areas of big data, cloud computing and networking, and distributed systems. His work was recognized by the 2016 Annual Best of Computing in ACM Computing Reviews. He is the recipient of the 2017 IEEE Systems Journal Annual Best Paper Award and other five Best Paper Awards from IEEE/ACM conferences. Prof. Guo was an Associate Editor of IEEE TPDS and an IEEE ComSoc Distinguished Lecturer. He is now on the editorial board of IEEE TCC, IEEE TETC, IEEE TSUSC, IEEE TGCN, IEEE Network, etc. Prof. Guo also served as General and TPC Chair for numerous IEEE conferences. He currently serves as a Director and Member of the Board of Governors of ComSoc. He is Fellow of IEEE.