A new approach for cross-silo federated learning and its privacy risks

1st Michele Fontana
University of Pisa
Pisa, Italy
michele.fontana@phd.unipi.it

2nd Francesca Naretto
Scuola Normale Superiore
Pisa, Italy
francesca.naretto@sns.it

3tr Anna Monreale
University of Pisa
Pisa, Italy
anna.monreale@unipi.it

***Abstract*—Federated Learning has witnessed an increasing popularity in the past few years for its ability to train Machine Learning models in critical contexts, using private data without moving them. Most of the approaches in the literature are focused on mobile environments, where mobile devices contain the data of single users, and typically deal with images or text data. In this paper, we define HOLDA, a novel federated learning approach tailored for training machine learning models on data distributed over federated organizations hierarchically organized. Our method focuses on the generalization capabilities of the neural network models, providing a new mechanism for selecting their best weights. In addition, it is tailored for tabular data. We empirically test the performance of our approach on two different tabular datasets, showing excellent results in terms of performance and generalization capabilities. Then, we also tackle the problem of assessing the privacy risk of users represented in the training data. In particular, we empirically show, by attacking the HOLDA models with the Membership Inference Attack, that the privacy of the users in the training data may have high risk.**

***Index Terms*—Federated Learning, Privacy risk assessment**

I. Introduction

Machine learning (ML) models can solve complex tasks, provided a good training set on which they can learn. When data are human-generated, collecting them and moving them from a site to another might be difficult, due to privacy regulations, like the GDPR[1]. An interesting solution is Federated Learning (FL), proposed in 2016 by McMahan et al. [10]: a ML technique that trains a model across multiple devices, without moving their data and hence, keeping their data private. In the last few years, the FL approach has become popular: it has been adopted in query suggestion [23], next word prediction [7] and medical imaging [8]. There are two main approaches in FL: *cross-device*, in which the clients are mobile devices, and *cross-silo*, which deals with organizations. For this work, we tackle the problem of *cross-silo* FL. In the literature, there are many applications and variants of FL for cross-device settings, while the cross-silo one has been analyzed less. Moreover, the majority of the works focus on images and text data, while our work is focused on tabular data. To the best of our knowledge, no work applies the FL approach on tabular data.

In this paper, we define HOLDA (Hierarchical crOss-siLo feDerated Averaging), a FL methodology tailored for training a ML model in the context of federated organizations that are hierarchically organized, in which each organization owns an imbalanced tabular dataset. An example is the hospitals setting: in this case, each hospital has its own local dataset that cannot be moved for privacy reasons. For each hospital, the data may be imbalanced and depend on the visits they specialize in and the area where the hospital is located. As a result, an individual ML model trained only on data from a single hospital might be overfitted against the diseases it sees most at that hospital and hence not be able to recognize other diseases. For this reason, it is crucial to develop models that can generalize very well, and to this end, we propose to exploit FL. Regarding the hierarchical FL approach, Liu et al. theorized a hierarchical FL, tailored for mobile devices: they interpose a layer of proxies between the local clients, and the central server [17]. In particular, in their approach each local client and the intermediate servers share the parameters of their last local model, obtained at the end of the local/intermediate learning phase. Differently from their work, we tailor our approach for achieving good generalization capabilities. Hence, in the training phase of our local and intermediate models, we do not share the last model, but the one which *generalizes* better on the local data.

After the definition of HOLDA, we also tackle the problem of the privacy of the users in its training data. In fact, in [21] Shokri et al. proposed the Membership Inference Attack (MIA): a privacy attack tailored for black-box models that is able to infer the membership of a record to the black-box's training set. In [19], the authors stated that the black-box MIA may not be applicable in FL contexts, due to the different training phases and the good generalization capabilities of the final models. However, we empirically show that attacking FL models with the black-box MIA puts the users' privacy in the training dataset at significant risk, with attacks that achieve a precision of 87%.

Wrapping up, our paper presents two contributions:

1. We define and test a novel FL learning approach, called HOLDA, tailored for federated organizations that are hierarchically organized and deals with tabular data. Our approach focuses on the generalization capabilities of the intermediate and local models;
2. We empirically assessed the privacy risk of the users in the training data during the HOLDA. We exploited the

[1]EU GDPR can be found at the following link: http://bit.ly/1TlgbjI.

Membership Inference Attack: it discerns if a record was part of the training dataset or not.

The paper is organized as follows. Section II discusses the related work. In Section III we introduce the definitions needed for presenting our contribution. In Section IV, we present our HOLDA approach and the privacy risk assessment in this setting. Lastly, Section V reports the experimental validation while Section VI concludes the work and discusses future research directions.

II. Related Work

Federated Learning Neural networks can benefit from large datasets to produce a model with good generalization performance. However, nowadays, data are often massively distributed over different parties, such as mobile devices and organizations. Due to privacy restrictions and regulations, like the GDPR, these distributed datasets cannot be transmitted to a central server to train a unique ML model. To address this issue, in 2016 McMahan et al. [10] proposed the idea of FL, which allows a set of distributed parties to train a global model while keeping their data private. In the last few years, this new paradigm has been adopted to solve a wide variety of tasks, such as query suggestion [23], next word prediction [7] and medical imaging [8], most of them focusing on image and text data. To the best of our knowledge, no one has still applied the FL framework on tabular data.

Depending on the clients nature, we can identify two main scenarios: the *cross-device* and *cross-silo* FL. In the first case the clients are mobile devices (smartphones, laptops, sensors), while in the second are organizations (hospitals, banks, data centers). In their work Li et al. [13] carefully analyzed these two settings from several points of view, such as the scale of the federation, the computational power and the stability of the parties. One important aspect that differentiates the two scenarios is the clients' statefulness. While in the cross-device setting, the parties are supposed to be stateless, since they are supposed to be able to leave the federation at any time due to connection problems, the clients in the cross-silo scenario are assumed to be stateful, that is they can save intermediate states of the training process and reuse them later on if needed.

Another key factor in FL is the architecture employed. The most widely used architecture is the *centralized* one, where a central server orchestrates the communication between the clients and the server itself. However, as in every centralized architecture, the server represents a single point of failure of the whole system. Moreover, Lian et al. [16] demonstrated that in the cross-device setting, the server can easily become a bottleneck due to the high communication traffic jam on the central node. This is because all nodes need to communicate with it concurrently and iteratively. Therefore, the performance will be significantly degraded when the network bandwidth is low. To reduce the workload on the central server, Liu et al. [17] suggested adopting a *hierarchical* architecture, putting one or more intermediate layers made up of proxies between the server and the participants.

A hot research topic is about developing efficient training algorithms in a federated environment. The first algorithm developed in this context is the Federated Averaging, or FedAvg, proposed by McMahan et al. in 2017 [18]. In each round of FedAvg, the updated local models of the parties are transferred to the server, which then further aggregates the local models to update the global model. In this work, we present a variant of this algorithm and hence we exhaustively describe it in Section III-A. As of its convergence, many studies revealed that the main factor that influences is the data distribution. While [18] empirically demonstrated that FedAvg is able to produce good performance models if the federated datasets are identically distributed, Li et al. [15] showed that the algorithm may be unstable and even diverge if it has to deal with non-IID data. There have been some studies trying to address the non-IID issue in the local training phase [12]. For example, FedProx [14] directly limits the local updates by $l2$-norm distance, while SCAFFOLD [9] corrects the local updates via variance reduction.

Data privacy in federated learning ML models learn on data. After the training, even if the data are not exposed, querying the model may still leak sensitive information about the people in the training dataset [1], [4], [21]. In the context of data privacy, to mathematically assess the privacy risk of the users in a dataset, a *privacy risk assessment* process has to be performed. Depending on the results of the privacy risk assessment, a *privacy protection* technique on the data can be applied to protect the users from malicious adversaries. Examples of privacy protection techniques are differential privacy or k-anonymity [22]. In the following, we present the state-of-the-art techniques in the context of privacy risk assessment since it is the main topic of this paper.

In [20], Pratesi et al. proposed PRUdence, a framework able to assess and then protect the privacy risk of the people in a dataset. In that setting, to quantify the privacy risk of people, the authors simulate a privacy attack on the dataset. Following this structure, during the last few years, there have been many privacy attacks against ML models: *membership inference attacks* (MIA) [21], in which the goal is to infer the membership of a given record to the training set of the black-box model; *reconstruction attacks* [4], [5], in which the goal of the attacker is to reconstruct one or more training samples and/or their respective training labels, *property inference attacks* [6], in which the goal is to extract information that the model learned unintentionally and hence are not explicitly encoded as features. As an example, property inference attacks may find out information such as the ratio of men and women in the training dataset. Clearly, this kind of attack may be used together with the membership inference attack or the reconstruction one to improve the knowledge of the adversary.

Privacy attacks have also been conducted on FL models. In particular, there is a white-box version of the MIA tailored for FL models [19]. In this paper the authors presented two white-box versions of the MIA to FL models: passive and active attack. Both of them exploited the privacy vulnerabilities of the stochastic gradient descent algorithm, which is the algorithm

used to train the neural networks. In this setting, the attacker knows all the loss functions, the gradients of the loss, the model's output and the one-hot encoding of the true label. The idea is that the distribution of the model's gradients on members of its training data versus non members is likely to be distinguishable since the vector is significantly large and cannot properly generalize over the training data. The actual attack consists of a convolutional neural network and a fully connected neural network. The output of these two neural networks is then fed into a fully connected encoder, which outputs a single value, that is the embedding of the membership information. In the case of federated learning, the attacker can observe each feature multiple times and stack them before using the actual attack component. In this paper the authors also mentioned that the original `MIA`, tailored for general ML models, may not achieve good results in a FL context due to the good generalization capabilities of the structure. However, in our work we show that the original membership inference attack is able to infer the membership of people in a training dataset also in a FL scenario.

III. Background

A. Federated Averaging

Federated Averaging (`FedAvg`) is the first training algorithm proposed to train deep neural networks in a centralized federated environment [18]. In the following, we suppose to have a central server S, which orchestrates the work of a federation of C clients. The goal is to train a neural network $\mathcal{N}$ by executing a given set of epochs.

The procedure starts with the server that randomly initializes the neural network parameters w_0 and then it executes the specified training rounds. In the following, we refer to them as global iterations to distinguish them from the training rounds executed on the client-side, also called local iterations. A generic global iteration j can be split into four main phases: sending, local training, aggregation and evaluation phase. In the *sending* phase, the server samples a subset C_i of k clients and sends them w^j, that is the current global model's parameters, through the dedicated communication channels. Every client $c \in C_i$, after having received w^j, starts training it for E epochs on its private dataset, applying one classic optimizer, like SGD, Adam or RMSProp. The number of local epochs and the optimizer are user-defined parameters. Finally, the client c sends back to the server the updated model parameters w_c^{j+1}, ending the local training phase of the algorithm. It is worth saying that some versions of Federated Averaging add an additional step, where they apply model compression techniques to reduce the amount of data transferred over the network [11], [2], [3]. The simplest one is the Δ-*Compression*, which involves the computation of the quantity $\Delta w_c^{j+1} = w_c^{j+1} - w_c^j$, that is the difference between the values of the new model parameters and the received one. Clients exchange this new quantity with the central server. When the server ends gathering all the results from the clients, it performs the *aggregation* phase, where it computes the new global model parameters, w^{j+1} as $w^{j+1} = w^j + \sum_{c \in C_i} \frac{n_c}{n} \Delta w_c^{j+1}$, where n_c is the number of records in the client c's training set and $n = \sum_{c \in C_i} n_c$. Therefore, in the last phase, the *evaluation* one, the server evaluates the new global model w^{j+1} according to the chosen metrics.

The algorithm we presented above can only deal with centralized architectures, where the clients are directly connected to the main server. Liu et al. [17] proposed an extension of `FedAvg`, called *Hierarchical Federated Averaging* (`HierFedAvg`), which can be executed on systems that interpose one layer of proxies between the participants and the central node. The main idea is that, since a proxy is just an intermediate server, every proxy can execute `FedAvg` on the subset of clients directly connected to it and send back the results to the upper level of the hierarchy. More specifically, `HierFedAvg` executes $M_{server} \times M_{proxy}$ iterations, where M_{server} and M_{proxy} are the number of iterations performed by the server and by a single proxy, respectively. At a generic iteration $j \in [1, M_{server}]$, the server sends the global model parameters w^j to all the proxies. Therefore, the proxy p, after receiving the initial parameters, starts executing M_{proxy} rounds of `FedAvg`, where p acts as a server and the participants are all the clients under its responsibility. Thus, denoting with C_p the set of all clients connected to p, we have that the updated model parameters computed by p can be expressed as $w_p^{j+1} = FedAvg(w^j, p, C_p)$. Finally, the main server collects and further aggregates these intermediate results to get the new global model.

B. Membership Inference Attack

To assess the privacy risk of the people in the training datasets used in a FL model, we exploited the Membership Inference Attack (`MIA`) theorized in [21]. `MIA` is a privacy attack against general machine learning models, hence it is not tailored for a FL scenario. It is a black-box attack: the adversary can only query the model to obtain the prediction, and she/he does not know any other information about the model.

`MIA` attacks a black-box model, referred to $f_b(\cdot)$, with its private training dataset D_b^{train} composed by $(x^i, y^i)_b$. Technically, x_b^i is the input to the model and y_b^i is its true label, which is a value among the ones in the set of possible classes, with dimension c_b. The output of the black-box model is a probability vector $\overline{y}_b$ of size c_b and the sum of all the values is one. To attack the black-box model, `MIA` defines an attack model $f_{attack}(\cdot)$: it is a machine learning model able to discern if a record was part of the training dataset or not. In practice, it is a binary classifier which predicts "yes" if the record was part of the training set, or "no" otherwise. $f_{attack}(\cdot)$ is trained on a dataset D_a^{train}, composed by $(x^i, y^i)_a$, where x_a is the input extracted from the record x. x_a is composed by the correct label of the record under analysis and the probability vector $\overline{y}_b$ queried from f_b, of length c_b; while y_a is the correct membership label and can be "in" or "out". The attack model $f_{attack}(\cdot)$ is a voting model, composed of c_b machine learning models: one for each output class of the black-box model under attack. The key factor in this attack

is the knowledge of the probability vector: given how the probabilities in $\overline{y}_b$ are distributed around the true value of the record, the attack model computes the membership probability $\Pr\{(x, y) \in D_b^{train}\}$, which is the probability that x belongs to the "in" class, i.e. it is part of the training set. To obtain the dataset D_a^{train}, on which the MIA model $f_{attack}(\cdot)$ is trained, the authors used *shadow models*. In fact, we are considering a black-box setting, hence we have no knowledge of the dataset used for training the black-box model. To overcome this limitation, they employed a set of k shadow models $f_s^i(\cdot)$: machine learning models trained to mimic the decisions of the black-box model $f_b(\cdot)$ we would like to attack. These shadow models are trained on $D_{s^i}^{train}$: these datasets are composed by $(x^i, y^i)_s$, in which x_s has the same format and similar distribution w.r.t. to the dataset employed to train the black-box model, while y_s is the predicted class obtained querying the black-box model. It is important to remark that the datasets employed for the training of the shadow models are disjoint from the unknown dataset used to train the black-box model.

IV. HOLDA AND ITS PRIVACY RISK

In this Section, we first propose HOLDA, our FL methodology, by providing a detailed description and its pseudo-code. Then, we describe the privacy risk assessment of the users in the training dataset by exploiting the MIA.

HOLDA. FL has witnessed increasing popularity in the past few years for its ability to train ML models in critical contexts, using private data without moving them. The majority of the works proposed focuses on mobile devices, in which every device contains the data of a single user, and deals with images or text data. In this work, we focus on FL for organizations that can also be applied on tabular data.

We define the *Hierarchical Cross-Silo Federated Averaging* (HOLDA) approach. *Cross-silo* means it is designed for settings in which the clients are organizations with data that cannot be moved from the organization's servers. Our approach is *hierarchical*: if needed, we interpose layers of proxies between the clients and the global server. The hierarchical FL approach (HierFedAvg) was already proposed for mobile devices in [17]. In this paper, we propose a novel hierarchical FL approach tailored for organizations. Our approach, HOLDA, significantly differs from HierFedAvg: in HierFedAvg, during the local training, each client and the intermediate servers share the parameters of their local model, after the last training epoch. Differently from the HierFedAvg, during the training phase of our local and intermediate models, HOLDA shares the model that better *generalizes* on the local data. We remark that this operation is possible because in a cross-silo setting the clients are assumed to be stateful.

In Figure 1, we report a detailed scheme of the training phase of HOLDA. In the picture, there is only a layer of intermediate servers, but we can have more layers without impacting the procedure. There are 3 kinds of participants: the global server S, a set of intermediate servers $I = \{I_1, \ldots, I_n\}$ and the local clients $C = \{C_1, \ldots, C_m\}$ (with $n < m$). Each party has its own neural network, with the same structure as the others. The process starts with the global server S, which randomly initializes the parameters w_s of its neural network. At this point, the server S starts the global training phase: a loop of N global iterations, in which at each iteration it calls the HOLDA function, presented in Algorithm 1. In the following, we refer to $U = \{u_1, \ldots, u_n\}$, where each u_i is a child node of a specific server that can be the global server or one of the intermediate ones.

Algorithm 1: HOLDA (w_s, E, U)

Input: w_s are the initial parameters, when $j = 1$
Input: E is the number of epochs
Input: U is the set of children nodes
Output: M_{best}, w_{best} the last best parameters and its metrics

1 **for** $j \leftarrow 1$ **to** E **do**
2 $\quad W, T = \textit{UpdateModel}(w_s^j, U)$;
3 $\quad w_s^{j+1} = \textit{Aggregation}(W, T, j)$;
4 $\quad M_{best}, w_{best} = \textit{Evaluation}(w_s^{j+1}, M_{best}, w_{best})$;

Once S calls the HOLDA procedure, for each epoch $j = 1, \ldots, E$ asks for the model updating to its children nodes starting from the parameters w_s^j (line 2). In case the child node is an intermediate server the recursive call to HOLDA starts with input the w_s^j parameters received by S. While in case the child node is a local client, it trains the model with the received parameters and sends back to the father node both *(i)* the updated weights, corresponding to the *best* model between all the previous (local or global) models and the new local model, and *(ii)* the number of records used for the local training. Note that, in our case the *best* model corresponds to the model that better *generalizes* on the local data.

Once gathered the updated weights W and the list of the number of local records T, each intermediate server (or the global server) applies the *Aggregation* function, described by Algorithm 2. Here, the weights of the received parameters are combined together, obtaining a new aggregated model corresponding to the $j + 1$-th iteration, having w_s^{j+1} as its parameters (line 2). We remark that $\Delta w_{u_k}^{j+1}$ is the difference between the parameters of the best model stored so far and the one received at the beginning of the iteration. Then, the *Evaluation* function (Algorithm 3) is applied. It *(i)* queries the children nodes to get their models evaluation M on the validation set (line 1), *(ii)* computes the average evaluation $\overline{M}$, and *(iii)* selects the best model (i.e., w_{best}) among the ones produced so far (lines 3-5).

After the N iterations of the global training are concluded, it is possible to apply a *fine tuning* of the models: we can train for one last time the models on the local data. This procedure is optional and allows for a better specialization of the model on the data under analysis. The fine-tuning operation may be done both at the client level and at the intermediate one: for the clients, the fine-tuning corresponds to a last training of the model, while for each intermediate server, the fine-tuning is executed by applying the HOLDA function on the sub-hierarchy

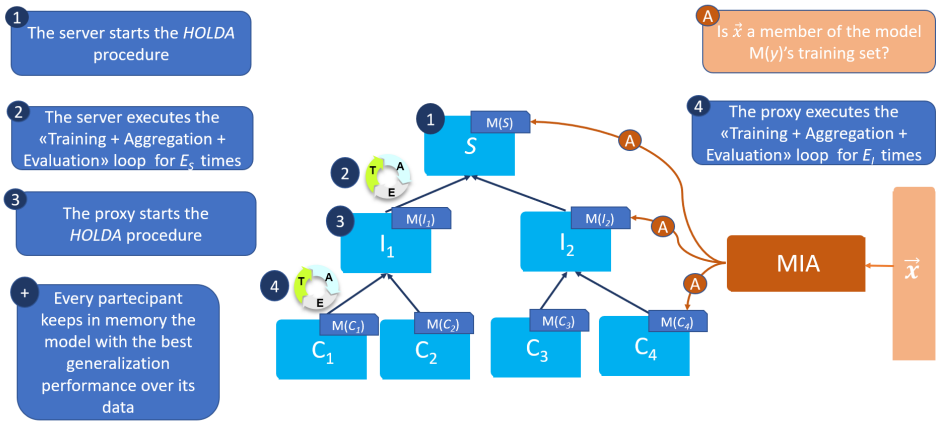

Fig. 1: Summary of our contribution. Taking as example the federation in the middle of the picture, we depict the main training steps performed by HOLDA as well as the attacks we conducted on the final models to evaluate the privacy risk of the users in the distributed training datasets.

Algorithm 2: Aggregation(W, T, j)

Input: $W = [w_{u_1}, \ldots, w_{u_n}]$, in which each w_{u_i} is a list of parameters from u_i
Input: $T = [t_{u_1}, \ldots, t_{u_n}]$, where t_{u_i} is the number of local records for u_i
Input: j is the id of the current iteration
Output: w_s^{j+1}, the updated list of parameters for s

$t = \sum_{k=1}^{n} t_{u_k}$;
$w_s^{j+1} = w_s^j + \sum_{k=1}^{n} \frac{t_{u_k}}{t} \Delta w_{u_k}^{j+1}$;

Algorithm 3: Evaluation(w_s, M_{best}, w_{best},U)

Input: w_s the parameters to validate
Input: M_{best} the metric's best value so far
Input: w_{best} the best parameter set so far
Input: U is the set of children nodes
Output: $M_{\text{best}}, w_{\text{best}}$, metrics and weights of the best model so far

$M = \textit{EvaluateModel}(w_s, U)$; // M is a list of metrics returned by U
$\overline{M} = \frac{1}{|U|} \sum_{m \in M} m$;
if $\overline{M} > M_{best}$ **then**
    $M_{\text{best}} = \overline{M}$;
    $w_{\text{best}} = w_s$;

that has its self as root node.

We remark that our methodology may be applicable also without considering a hierarchical setting: in this case, we have two layers, the global server and the clients, but we still select the model with the best generalization capabilities depending on the local data, as well as we have the optional fine-tuning operation.

The choice to define HOLDA, is dictated by the different context in which we want to use the FL approach. In fact, if in the context of mobile devices, the choice to select the last trained model can be successful, in the context of cross-silo devices, where clients have many data from similar types of users, the last model could be overfitted.

The context we thought of when developing this methodology is hospital organizations. In this setting, there are many hospitals, each with its own local dataset that cannot be moved for privacy reasons. For each hospital, the data is dependent on the visits they specialize in, the climate, and the type of work that is done in the area where the hospital is located. As a result, an individual ML model trained only on data from a single hospital might be overfitted against the diseases it sees most at that hospital and hence not be able to recognize other diseases when it encounters them. For this reason, it is very important to develop models that can generalize very well, and our solution is to use HOLDA, so that the ML algorithm can see examples of different types. In this setting, hospitals may be grouped geographically, hence creating intermediate layers of cities, regions, states, etc. Clearly, privacy constraints may be different depending on the kind of geographical group considered. Moreover, grouping in this way may help the ML models to specialize in the diseases encountered most in the geographical region under analysis. For these reasons, we considered a hierarchical context, in which we have different layers of generalization.

Privacy risk assessment on HOLDA. After the definition of HOLDA, we consider the privacy guarantees of the different ML models produced by our FL methodology. In [19], the authors suggested that the black-box MIA [21] is not going to be a powerful attack in the context of federated learning due to the fact the models generalize well. In addition, they propose a white-box MIA, tailored for FL models, which exploits notions available during the training phase. However, if we assume to be able to guarantee a secure exchange of information during the training phase, the white-box attack may not be performed. Instead, after the training, when the ML models are published,

the standard black-box MIA may still be applied. In particular, in the setting of HOLDA, several models can be published: the global model, a set of intermediate models and a set of local models. All of them could be attacked exploiting MIA. Thus, we propose a privacy risk assessment methodology that evaluates the privacy risk of each ML model produced level by level in our hierarchical schema.

Depending on the kind of attacked model, the simulation of the attack tries to infer the membership of different records. When dealing with local models, the attacker may infer the membership of the training records exploited during the training of the client under attack. As an example consider Figure 1. If we are attacking the client C_1, then the attacker is trying to reconstruct its training dataset, T_{C_1}. Regarding an intermediate model, it is trained considering all the training samples of all its clients. Hence, in this case the attacker tries to infer the membership of $T_{I_1} = \{T_{C_1}, ...T_{C_p}\}$, which is the union of all the training sets of its clients $C_1, ..., C_p$. Considering Figure 1 as an example, when the attacker attacks I_1, the training sets considered are the ones used for training client C_1 and C_2. This scheme applies recursively to all parent nodes, such as the global one or models from other intermediate layers, if any. By considering these different scenarios, our contribution is to empirically demonstrate that the MIA can actually expose the privacy of the users in the training datasets also in FL, even if the models show a good generalization capability.

Contribution Summary. Overall, our proposed approach can be summarized as follows:

1. We define HOLDA, a Hierarchical Cross-Silo FL approach, in which the key factors are:
   - a hierarchical setting, in which we allow for the possibility of layers of proxies between the global server and the local clients;
   - the weights exchanged are the ones that generalize better. The generalization capabilities are validated on the local data for the clients and on the mean of a metric score for the intermediate and global servers;
   - as last, a fine-tuning operation is available for both clients and intermediate servers. For the intermediate servers, the fine-tuning operation corresponds to a last HOLDA computation, while for the clients, it is a last training phase on the local data.
2. An empirical privacy risk assessment of the users in the training dataset of HOLDA. In practice, we attack the ML models produced with HOLDA using the black-box MIA.

V. Experiments

We experimentally validate: (i) the performance of HOLDA, our FL approach; and (ii) the power of the black-box MIA on HOLDA, against every model produced by the FL approach. We considered two datasets with different characteristics: a medical dataset and a mobility dataset[2,3]. We divided each dataset into training and test, using an $80 - 20$ splitting, stratifying it over the class labels distribution.

A. Medical Dataset Description

The medical dataset used to conduct our experiments is the publicly available *Texas-100* dataset[4] (from now on called TexHosp), provided by the Texas Health Service Department. It is a tabular dataset, which describes hospital stays in over 200 Texas hospitals. We consider the period from 2006 to 2009, extremes included. The dataset contains $5,426,184$ records and it has 208 features, the majority of them are categorical. The categorical features provide details about the causes of the injury, the main diagnosis made by the medical staff, and the surgical procedures the patient underwent. Moreover, the records also give information about the hospital (name, facilities) and the patient (gender, sex, age, home country) themselves. The task we want to solve is predicting the patient's main procedure. In our work, we focused on the 61 most frequent procedures. Analyzing the class distributions, we found out that the dataset is highly imbalanced: while the instances of the most represented class are over 500,000, there are just 10,000 records that are labeled as the least frequent class. As of the pre-processing phase, we discarded all those attributes that contained at least 90% of missing values, reducing the dataset features to 85. To conduct experiments in a FL environment, we split the data into smaller and disjoint chunks. Each chunk represents the private dataset stored into a client of the federation. We considered a federation made up by 11 clients; each of them corresponds to a Texas Public Health Region, or PHR. Therefore, the private dataset of client $c \in [1, 11]$ contains the records of all the hospitals that are located in the PHR c. For the hierarchical setting, we grouped each client into 4 areas on a geographical basis: Center, East, South and North West. Each area is then associated with a proxy in the hierarchical scenario. More precisely, Area Center is made up of the regions $2, 3, 7$, while Area East contains the PHR $4, 5, 6$. These are the two biggest areas of the whole federation, ($1,700,000$ and $1,400,000$ records in their private datasets). Regions 8 and 11 are in the Area South's group, with $899,108$ total records. The Area North West contains $1, 9$ and 10, with a total of $409,602$ records.

B. Mobility Dataset Description

The other data employed is a mobility dataset, referred to Mobility, extracted from GPS tracks of private vehicles in Tuscany (Italy) provided by Octo Telematics. We constructed a dataset in which the classes describe the mobility behaviors of the users. We selected trajectories from an area comprising 4 urban centers: Firenze, Prato, Pistoia and Livorno. We consider the period from 1st May to 31st May 2011, for a total of $17,756$ distinct vehicles. The trajectory points are generalized according to the geographical tessellation provided by the Italian National Statistics Bureau (ISTAT): each point is substituted with the centroid of the geographical cell to

[2]We performed the experiments on a server having an Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz, 16 cores and 64GB of RAM

[3]Code available upon acceptance

[4]https://www.dshs.texas.gov/thcic/hospitals/Inpatientpudf.shtm

which it belongs. We then removed redundant points, i.e., points mapped to the same cell at the same time, obtaining $4,185$ different locations. From these trajectories, we extracted a mobility profile for each user: in Table I all the extracted features are listed. At this point, we applied the k-means clustering algorithm to group people with a similar mobility behavior. We conducted an exhaustive search of the best k by using the *elbow method*. We selected $k = 6$. Hence, this mobility dataset has 6 classes.

C. FL experiments

In this section we present the prediction performance of HOLDA on the TexHosp dataset (both hierarchical and non hierarchical approach) and on the Mobility dataset (non hierarchical approach).

TexHosp. On this dataset, we applied 2 variants of HOLDA: a *non-hierarchical* approach, in which we only have 2 layers, i.e. the global server and the local clients; and a *hierarchical* approach, in which we added an intermediate layer of servers. Regarding the non-hierarchical approach, we obtained very good performance, reported in Table II, 5-cross validated. We considered a central server, which coordinates the work of a set of 11 clients, one for each hospital region. The model selected is a feedforward neural network with two hidden layers, made up by 800 and 400 neurons, respectively. The activation function is ReLu and we put a dropout layer after the last hidden layer to regularize the model (with rate 0.2). Lastly, we applied the softmax function in the output layer. To train the model we performed at most $1,000$ global training phases. We decided to apply the early stopping technique, with a patience of 10 epochs as our main stopping condition. As of the local training, we executed 3 rounds, using the Adam optimizer with a learning rate of $1e^{-4}$ and a batch size of 128. As local loss we used the CrossEntropy. We precise that the global loss is computed as the weighted average of all the local losses that the server gets from the clients or intermediates. Specifically, the weighting strategy is identical to the one of FedAvg algorithm, with the exception that we select the weights of the model that *generalizes* better on the local data. Lastly, we applied the *fine tuning* phase, presented in IV. On average, a complete training phase of this model requires 14.4 hours.

In [16], the authors demonstrated that in the FedAvg setting, with only 2 layers, the server can become a bottleneck, due to the high communication traffic jam. To reduce the workload on the central server, Liu et al. [17] suggested HierFedAvg, which has an intermediate layer made up of proxies between the server and the participants. Since the TexHosp dataset is composed by a great number of local servers, we decided to apply also the HOLDA with the *hierarchical* level. We grouped the 11 clients in 4 geographical areas: Center, East, South and North West. In this setting we applied the same parameters configuration as in the case before, but we added 3 intermediate training rounds coordinated by the intermediate servers on their clients. Also in this case, we applied the *fine tuning* operation at the end. The results of the hierarchical HOLDA are reported in Table III. On average, a complete training phase of this model requires 28.8 hours.

Notation	Description	Notation	Description
V	visits	$\overline{V}$	daily visits
D_{max}	max distance	D_{sum}	sum distances
D_{max}^{tot}	max distance over total max distance for a user	$\overline{D}_{sum}$	D_{sum} per day
D_{max}^{trip}	D_{max} over area	$Locs$	distinct locations
$Locs_{ratio}$	$Locs$ over area	R_g	radius of gyration
E	mobility entropy	E_i	location entropy
U_i	individuals per location	U_i^{ratio}	U_i over individuals
w_i	location frequency	w_i^{pop}	w_i over the total frequency of location i
$\overline{w}_i$	daily location frequency	PT_j	Path time per user

TABLE I: Mobility features of the individual mobility profile.

Overall we can notice that the prediction performance of the models are good both in the non-hierarchical setting (Table II) as well as the hierarchical one (Table III). However, the hierarchical HOLDA shows slightly better performance, especially for the local models (up to 4 percentual points more than the previous scenario) and we have a hierarchical structure that may be useful in some organization settings.

In both cases, the models generalize really well. This is especially true for the hierarchical and the global models, in which the difference between the performance on the training and the validation test differs by one or two percentage points at most.

Mobility. On this dataset we only applied the *non-hierarchical* HOLDA. We considered one global model and 3 clients ones, one for each traffic region considered: Firenze, Prato and Pistoia, Livorno. We selected a feedforward neural network with two hidden layers, 300 neurons for the first layer and 100 neurons for the second one. We chose the ReLU activation function and we applied a softmax to the output layer. To regularize the model, we applied a dropout after the last hidden layer (rate 0.1). For the local models, we choose Adam optimizer with a learning rate of $1e^{-4}$ and a batch size of 128. As local loss, we used the CrossEntropy. Also, in this setting, the global loss is computed as the weighted average of all the local losses that the server gets from the clients or proxies. We trained the model for at most $1,000$ global iterations, with early stopping (patience of 10), and 3 training epochs on the local clients. Also in this case, we applied the *fine tuning* operation on the client models. The results obtained from the application of *non-hierarchical* HOLDA are shown in Table IV.

We can notice good prediction performance both in the local models as well as the global one, even if the global model suffers from a considerable loss in performance w.r.t. the client ones (an average loss of 0.12). However, this is due to the data we are using: the mobility of the 3 regions considered is quite different. Moreover, on the clients, we applied the fine-tuning, making the local models tailored for their local data. Lastly, we can notice that both the local and the global model show good generalization capabilities.

D. Membership Inference Attack experiments

After having presented the excellent results obtained in the application of HOLDA, we present the results of the MIA. We remark that this privacy attack exploits the probability vector

$Kind$	Model	Metric	Training	Validation	$Kind$	Model	Metric	Training	Validation
G	All	F_1	0.78 (0.01)	0.76 (0.01)	L	Reg1-NW	F_1	0.85 (0.01)	0.81 (0.01)
		$Prec$	0.80 (0.01)	0.79 (0.01)			$Prec$	0.86 (0.01)	0.82 (0.01)
		Rec	0.78 (0.00)	0.76 (0.01)			Rec	0.84 (0.01)	0.80 (0.00)
L	Reg2-C	F_1	0.89 (0.01)	0.84 (0.01)	L	Reg3-C	F_1	0.83 (0.01)	0.81 (0.01)
		$Prec$	0.89 (0.01)	0.85 (0.01)			$Prec$	0.84 (0.01)	0.82 (0.01)
		Rec	0.88 (0.01)	0.84 (0.01)			Rec	0.85 (0.01)	0.82 (0.01)
L	Reg4-E	F_1	0.85 (0.01)	0.82 (0.01)	L	Reg5-E	F_1	0.87 (0.01)	0.83 (0.00)
		$Prec$	0.86 (0.01)	0.84 (0.01)			$Prec$	0.88 (0.01)	0.84 (0.01)
		Rec	0.85 (0.01)	0.82 (0.01)			Rec	0.87 (0.01)	0.82 (0.00)
L	Reg6-E	F_1	0.82 (0.01)	0.80 (0.00)	L	Reg7-C	F_1	0.83 (0.01)	0.81 (0.01)
		$Prec$	0.84 (0.01)	0.82 (0.01)			$Prec$	0.82 (0.01)	0.83 (0.01)
		Rec	0.81 (0.01)	0.80 (0.00)			Rec	0.85 (0.01)	0.80 (0.00)
L	Reg8-S	F_1	0.84 (0.01)	0.82 (0.01)	L	Reg9-NW	F_1	0.88 (0.01)	0.82 (0.00)
		$Prec$	0.86 (0.00)	0.84 (0.01)			$Prec$	0.89 (0.01)	0.83 (0.00)
		Rec	0.83 (0.00)	0.81 (0.01)			Rec	0.88 (0.01)	0.82 (0.00)
L	Reg10-NW	F_1	0.85 (0.01)	0.81 (0.01)	L	Reg11-S	F_1	0.84 (0.01)	0.82 (0.01)
		$Prec$	0.86 (0.01)	0.83 (0.01)			$Prec$	0.85 (0.01)	0.83 (0.01)
		Rec	0.85 (0.01)	0.81 (0.01)			Rec	0.83 (0.01)	0.81 (0.01)

TABLE II: Performance of the FL of the models trained with HOLDA on the TexHosp dataset in the non-hierarchical setting. In the $Kind$ column we show the type of the model: global (G), intermediate (I), local (L). In the *Model* column we identify the region and area corresponding to the model. We adopt the following format: Reg$X$-Y, where $X \in [1, 11]$ is the region's identifier and Y represents its geographical area: Center (C), East (E), South (S), North West (NW). The results reported are the mean and the standard deviation over a 5-fold cross validation.

$Kind$	Model	Metric	Training	Validation	$Kind$	Model	Metric	Training	Validation
G	All	F_1	0.80 (0.04)	0.78 (0.00)	I	C	F_1	0.83 (0.01)	0.80 (0.00)
		$Prec$	0.82 (0.00)	0.80 (0.00)			$Prec$	0.85 (0.01)	0.82 (0.00)
		Rec	0.79 (0.00)	0.77 (0.00)			$Recall$	0.83 (0.01)	0.79 (0.00)
I	E	F_1	0.83 (0.01)	0.79 (0.00)	I	S	F_1	0.86 (0.01)	0.82 (0.00)
		$Prec$	0.85 (0.01)	0.81 (0.00)			$Prec$	0.87 (0.01)	0.83 (0.00)
		Rec	0.83 (0.01)	0.79 (0.00)			Rec	0.86 (0.01)	0.81 (0.00)
I	NW	F_1	0.89 (0.01)	0.81 (0.00)	L	Reg1-NW	F_1	0.88 (0.01)	0.84 (0.00)
		$Prec$	0.89 (0.01)	0.82 (0.00)			$Prec$	0.89 (0.01)	0.84 (0.00)
		Rec	0.90 (0.01)	0.80 (0.01)			Rec	0.88 (0.01)	0.83 (0.00)
L	Reg2-C	F_1	0.93 (0.01)	0.85 (0.00)	L	Reg3-C	F_1	0.86 (0.01)	0.82 (0.00)
		$Prec$	0.93 (0.01)	0.86 (0.00)			$Prec$	0.87 (0.01)	0.83 (0.00)
		Rec	0.93 (0.01)	0.85 (0.00)			Rec	0.86 (0.01)	0.82 (0.00)
L	Reg4-E	F_1	0.89 (0.01)	0.84 (0.00)	L	Reg5-E	F_1	0.91 (0.01)	0.84 (0.00)
		$Prec$	0.90 (0.01)	0.85 (0.00)			$Prec$	0.92 (0.01)	0.85 (0.00)
		Rec	0.89 (0.01)	0.84 (0.00)			Rec	0.91 (0.01)	0.84 (0.00)
L	Reg6-E	F_1	0.86 (0.01)	0.82 (0.00)	L	Reg7-C	F_1	0.86 (0.01)	0.82 (0.00)
		$Prec$	0.88 (0.01)	0.83 (0.00)			$Prec$	0.88 (0.01)	0.84 (0.00)
		Rec	0.86 (0.01)	0.81 (0.00)			Rec	0.85 (0.01)	0.82 (0.00)
L	Reg8-S	F_1	0.89 (0.01)	0.84 (0.00)	L	Reg9-NW	F_1	0.88 (0.00)	0.84 (0.00)
		$Prec$	0.90 (0.01)	0.85 (0.00)			$Prec$	0.88 (0.00)	0.85 (0.00)
		Rec	0.88 (0.01)	0.83 (0.00)			Rec	0.89 (0.00)	0.84 (0.00)
L	Reg10-NW	F_1	0.88 (0.01)	0.84 (0.00)	L	Reg11-S	F_1	0.88 (0.01)	0.84 (0.00)
		$Prec$	0.89 (0.01)	0.85 (0.00)			$Prec$	0.90 (0.01)	0.85 (0.00)
		Rec	0.87 (0.01)	0.83 (0.00)			Rec	0.88 (0.01)	0.83 (0.00)

TABLE III: Performance of the FL of the models trained with HOLDA on the TexHosp dataset in the hierarchical setting. In the column $Kind$ we show the type of the model: global (G), intermediate (I), local (L). In the *Model* column we identify the region and area corresponding to the model. We adopt the following format: Reg$X$-Y, where $X \in [1, 11]$ is the region's identifier and Y represents its geographical area: Center (C), East (E), South (S), North West (NW). The results reported are the mean and the standard deviation over a 5-fold cross validation.

$Kind$	Model	Metric	Training Value	Validation Value	$Kind$	Model	Metric	Training Value	Validation Value
G	All	F_1	0.84 (0.01)	0.81 (0.01)	L	Firenze	F_1	0.98 (0.01)	0.96 (0.01)
		$Prec$	0.85 (0.01)	0.83 (0.01)			$Prec$	0.98 (0.01)	0.96 (0.01)
		Rec	0.84 (0.00)	0.81 (0.01)			Rec	0.98 (0.01)	0.96 (0.00)
L	Livorno	F_1	0.96 (0.02)	0.93 (0.01)	L	Prato	F_1	0.98 (0.01)	0.96 (0.00)
		$Prec$	0.96 (0.01)	0.93 (0.01)			$Prec$	0.98 (0.01)	0.97 (0.01)
		Rec	0.96 (0.01)	0.93 (0.01)			Rec	0.98 (0.00)	0.96 (0.00)

TABLE IV: Performance of the HOLDA training on the Mobility dataset. The *Model* column identifies the model trained on the data belonging to the specific city. The results reported are the mean and the standard deviation over a 5 fold cross validation.

$Kind$	Model	Metric	RF	$Kind$	Model	Metric	RF
G	All	$Prec$	0.72	I	C	$Prec$	0.71
		Rec	0.76			$Recall$	0.75
I	E	$Prec$	0.72	I	S	$Prec$	0.81
		Rec	0.76			Rec	0.75
I	NW	$Prec$	0.78	L	Reg1-NW	$Prec$	0.83
		Rec	0.76			Rec	0.80
L	Reg2-C	$Prec$	0.82	L	Reg3-C	$Prec$	0.74
		Rec	0.80			Rec	0.78
L	Reg4-E	$Prec$	0.83	L	Reg5-E	$Prec$	0.82
		Rec	0.85			Rec	0.87
L	Reg6-E	$Prec$	0.75	L	Reg7-C	$Prec$	0.83
		Rec	0.76			Rec	0.77
L	Reg8-S	$Prec$	0.82	L	Reg9-NW	$Prec$	0.83
		Rec	0.76			Rec	0.88
L	Reg10-NW	$Prec$	0.83	L	Reg11-S	$Prec$	0.83
		Rec	0.81			Rec	0.79

TABLE V: Performance of MIA on the models trained on the TexHosp dataset in the hierarchical setting. In the column $Kind$ we show the type of the model: global (G), intermediate (I), local (L). In the *Model* column we identify the region and area corresponding to the model. We adopt the following format: Reg$X$-Y, where $X \in [1, 11]$ is the region's identifier and Y represents its geographical area: Center (C), East (E), South (S), North West (NW).

Kind	Model	Metric	RF	NN	*Kind*	Model	Metric	RF	NN
G	All	$Prec$	0.84	0.84	L	Firenze	$Prec$	0.87	0.87
		Rec	0.84	0.64			Rec	0.94	0.88
L	Livorno	$Prec$	0.87	0.87	L	Prato	$Prec$	0.87	0.87
		Rec	0.92	0.89			Rec	0.93	0.90

TABLE VI: Performance of MIA on the models trained on the Mobility dataset. The *Model* column identifies the model trained on the data belonging to the specific city. The *RF* and *NN* columns show, respectively, the attack performance when Random Forest or Neural Network are used as shadow models.

information to determine if a record was part of the training dataset or not. Hence, in a context like HOLDA, in which the models generalize really well and the weights, during the training, are updated depending on other models, this attack may not be satisfactory [19]. However, in the following, we empirically demonstrate that this is not the case: MIA is able to attack the Fl models of both datasets, obtaining results that are alarming for the privacy of the users in the training datasets.

TexHosp. For the training of MIA, we refer to the original paper [21]: we considered the scenario in which the attacker may have access to some data that is similar to the target model's training data. Given a black-box model $f_b(\cdot)$, trained on D_b^{train}, we considered $D_{s^i}^{train}$, composed by n records, disjoint from D_b^{train}, to train the $k = 6$ shadow models $f_s^i(\cdot)$. The parameter n, which is the number of records considered for the training of the privacy attack, depends on the kind of models we are trying to attack: for the local models, we selected 40% of the test set, while for the intermediate and the global models, which are trained depending on the local training datasets under them, we selected $20,000$ records equally divided among the regions that refer to them. The shadow models mimic the behavior of $f_b(\cdot)$, hence for creating $D_{s^i}^{train}$ we queried $f_b(\cdot)$ to obtain the labels on which train the f_s^i. The shadow models are Random Forests (RF). We performed a grid search to find the best set of values for the hyper parameters, selecting 100 as the maximum number of trees, whose maximum depth is equal to 15 and minimum sample split at 5. After the training of the shadow models, we were able to create the dataset D_a^{train}, on which train the attack $f_{attack}(\cdot)$. For the attack model, we trained a set of RF, one for each output class of the black-box model $f_b(\cdot)$[5]. In Table V, we report the results obtained by attacking all the models derived from the TexHosp dataset. We remark that we report the results of the "in" class since in this setting we are interested into understand how many users actually in the training dataset are identified. Hence, our focus is on the precision of the "in" class (the proportion of users that the attack classified as "in" and were truly "in") and on the recall of the "in" class (the proportion of users correctly identified as "in" w.r.t. the ones that were "in" but the model classified as "out") of our attack. From these results, we can clearly see that all the models, if attacked using MIA, put the privacy of their users at risk. In particular, the models of the local regions put the users in the training dataset more in danger w.r.t. the hierarchical and global models. On average, the attack precision of the local models is 0.81, while for the hierarchical models it drops to 0.75 and for the global model is slightly less, 0.72. Regarding the recall, we observe a similar pattern: the local models have an average recall of 0.80, while the hierarchical models drop to 0.75. However, the global model shows a recall of 0.76, highlighting the stability of the attack.

[5]Their configuration is: 100 trees, whose maximum depth is equal to 15 and minimum sample split at 5

Mobility. For this dataset, we applied the same structure of the attack proposed for the TexHosp one. We considered $D_{s^i}^{train}$, composed by n records, disjoint from D_b^{train}, to train the $k = 6$ shadow models f_s^i. n is the number of records considered. Also in this case, for training the attack on the local models, we considered 40% of the test set, while for the hierarchical and global models we considered a set of $20,000$ records equally divided among the regions that refer to them. For this dataset, we considered 2 variants of the shadow models f_s^i: a set of Random Forests (RF) and one of Neural Networks (NN). For the RF, we conducted a grid search, selecting 100 maximum trees, maximum depth 15 and minimum sample split 5. Regarding the NN, we considered the same structure of the black-box model we are trying to attack: a feedforward neural network with 2 hidden layers. The result of attacking the Mobility dataset are reported in Table VI. In this case we can note an alarming situation for the privacy of the users in the training dataset, even more than in the case of TexHosp. This is particularly true when we consider the attack in which we employ the RF as shadow models. In fact, even if the "in" precision of the attack is the same for RF and NN (0.87 for the local models and 0.84 for the global models), the recall is significantly different. In the case of NN, local models are attacked easily: they have an average recall of the attack of 0.89, while the global model has 0.64, which is a particularly low value. Regarding the RF, instead, we achieve an average recall of 0.93 for the local models and 0.84 for the global ones.

VI. Conclusions

In this paper we presented HOLDA, a novel Federated Learning approach tailored for training machine learning models in case of federated organizations, which are hierarchically organized. The proposed approach is focused on the generalization capabilities of the learned models and is based on the selection of their best weights. Then, we also analyzed the privacy risk of the machine learning models trained with this novel Federated Learning approach. We simulated the attack of the different machine learning models that could be exposed with our schema (local models, intermediate models and global model) using the Membership Inference Attack. A wide experimentation showed good predictive performance of the machine learning models learned by our FL method, but also a level of privacy risk that is not negligible. The results

also highlighted that the intermediate and global models lead to a lower privacy risk with respect to the local ones.

We plan to investigate the proposed FL approach in different contexts characterized by different types of data (images, time series, texts), to assess the privacy risk also in those contexts, and to define suitable approaches for privacy mitigation.

Acknowledgments

This work is supported by the European Union – Horizon 2020 Program under the scheme "INFRAIA-01-2018-2019 – Integrating Activities for Advanced Communities", Grant Agreement n.871042, "SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics" (http://www.sobigdata.eu), TAILOR (G.A. 952215) and HumanE-AI-Net (G.A. 952026).

REFERENCES

[1] Al-Rubaie, M., Chang, J.M.: Privacy-preserving machine learning: Threats and solutions. IEEE Security Privacy **17** (2019)

[2] Alistarh, D., Grubic, D., Li, J., Tomioka, R., Vojnovic, M.: QSGD: communication-efficient SGD via gradient quantization and encoding. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA (2017)

[3] Caldas, S., Konečný, J., McMahan, H.B., Talwalkar, A.: Expanding the reach of federated learning by reducing client resource requirements. CoRR (2018)

[4] Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. CCS '15 (2015)

[5] Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., Ristenpart, T.: Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In: Proceedings of the 23rd USENIX Conference on Security Symposium. SEC'14 (2014)

[6] Ganju, K., Wang, Q., Yang, W., Gunter, C.A., Borisov, N.: Property inference attacks on fully connected neural networks using permutation invariant representations. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. CCS '18 (2018)

[7] Hard, A., Rao, K., Mathews, R., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., Ramage, D.: Federated learning for mobile keyboard prediction. CoRR (2018)

[8] Kaissis, G.A., Makowski, M.R., Rückert, D., Braren, R.F.: Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence **2**(6), 305–311 (2020)

[9] Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U., Suresh, A.T.: SCAFFOLD: stochastic controlled averaging for on-device federated learning. CoRR (2019)

[10] Konečný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. CoRR (2016)

[11] Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. CoRR (2016)

[12] Li, Q., He, B., Song, D.: Model-contrastive federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10713–10722 (June 2021)

[13] Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., He, B.: A survey on federated learning systems: Vision, hype and reality for data privacy and protection. arXiv e-prints pp. arXiv–1907 (2019)

[14] Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. In: Dhillon, I.S., Papailiopoulos, D.S., Sze, V. (eds.) Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020. mlsys.org (2020)

[15] Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of fedavg on non-iid data. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020)

[16] Lian, X., Zhang, C., Zhang, H., Hsieh, C., Zhang, W., Liu, J.: Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 5330–5340 (2017)

[17] Liu, L., Zhang, J., Song, S., Letaief, K.B.: Client-edge-cloud hierarchical federated learning. In: 2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7-11, 2020. pp. 1–6. IEEE (2020)

[18] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, X.J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR (2017)

[19] Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. 2019 IEEE Symposium on Security and Privacy (SP) (2019)

[20] Pratesi, F., Monreale, A., Trasarti, R., Giannotti, F., Pedreschi, D., Yanagihara, T.: Prudence: a system for assessing privacy risk vs utility in data sharing ecosystems. Transactions on Data Privacy **11**(2), 139–167 (2018)

[21] Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP) (2017)

[22] Torra, V.: Data Privacy: Foundations, New Developments and the Big Data Challenge. Springer Publishing Company, Incorporated, 1st edn. (2017)

[23] Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., Beaufays, F.: Applied federated learning: Improving google keyboard query suggestions. CoRR (2018)