Partially-federated learning: A new approach to achieving privacy and effectiveness

https://doi.org/10.1016/j.ins.2022.10.082

Partially-Federated Learning: a new approach to achieving privacy and effectiveness

Marco Fisichella[a], Gianluca Lax[b], Antonia Russo[b]

[a]*L3S Research Center, Leibniz University of Hannover, Appelstr. 9a, 30167 Hannover, Germany*
[b]*DIIES Dept., University Mediterranea of Reggio Calabria, Via Graziella, Località Feo di Vito, 89122 Reggio Calabria, Italy*

Abstract

In Machine Learning, the data for training the model are stored centrally. However, when the data come from different sources and contain sensitive information, we can use federated learning to implement a privacy-preserving distributed machine learning framework. In this case, multiple client devices participate in global model training by sharing only the model updates with the server while keeping the original data local. In this paper, we propose a new approach, called *Partially-federated learning*, that combines machine learning with federated learning. This hybrid architecture can train a unified model across multiple clients, where the individual client can decide whether a sample must remain private or can be shared with the server. This decision is made by a *privacy module* that can enforce various techniques to protect the privacy of client data. The proposed approach improves the performance compared to classical federated learning.

Keywords: Machine Learning, k-anonymity, l-diversity, Distributed databases, Collaborative learning

1. Introduction

The recent digital transformation in mobile networks (e.g., 5G) and computing has driven the growth of Internet users, connectivity, and IoT devices, as well as the demands on network functions and applications. Several new devices are launched each year with more capabilities and intelligence. As reported in the Cisco Annual Report (2018-2023) [7], the average number of devices and connections per capita will increase from 2.4 in 2018 to 3.6 in 2023, resulting in billions of devices in the world. However, mobile devices have limited computing power and battery capacity, making it infeasible or too resource intensive to manage large amounts of data locally, even if self-produced, and perform elaborate tasks. One solution is to move all or part of the data and tasks to a computing server.

This is done, for example, in standard deep learning techniques, where the data for model training are stored centrally, and each participant uploads its data to a single place. However, data may contain sensitive information that users do not want to disclose, such as gender, salary, health status, political orientation, and browsing history. Therefore, transferring these data to a server may violate the privacy of customers, and the concerns of individual subscribers may be greater than the benefits of deep learning services. In addition, Western legal frameworks are emphasizing increasingly strong data protection, with the Council of the European Union Commission activating the General Data Protection Regulation [28] in 2016.

To address this issue, researchers at Google have proposed a privacy-preserving distributed machine learning system, called Federated Learning, to train models from decentralized consumer data on IoT devices without

violating privacy [16]. Inspired by this framework, multiple client devices participate in training a global model while keeping the original data local. In this way, the utility of the data is well preserved while the data remains local. By definition, raw participant data does not leave client devices in federated learning environments, and clients only share model updates with the server.

The literature is rich with proposals to protect user privacy in federated learning environments, and these proposals fall into three main categories, namely model aggregation [27], homomorphic encryption [6, 12], and differential privacy [2, 9].

Federated learning solves the privacy problem [5, 14], but has the side effect of shifting the computational load to the device side.

This paper proposes a new approach called *Partially-federated learning* that combines the classical centralized computing framework for machine learning with the new federated learning framework. To the best of our knowledge, this is the first attempt in the scientific community to create such a hybrid architecture [39]. The intuition behind our proposal is based on the fact that client-side data contains two types of information: 1) confidential information, if it is classified as sensitive information; 2) non-confidential information, if it can be released without risk to the user's privacy. Since not all data stored on clients could violate privacy, we introduce the *privacy module* on the client side in our system. This module can implement any privacy criterion to determine which data can be shared with the server while respecting the privacy requirements. In our work, we consider the two most popular privacy criteria, namely k-anonymity and l-diversity. The concept

of k-anonymity is based on the fact that each tuple is indistinguishable from at least $k-1$ other records within the dataset [33]. The l-diversity approach limits the risk of identification by mandating that the sensitive attribute must have a minimum number of distinct values within each equivalence class, where an equivalence class is a set of identical quasi-identifiers[1] attributes sources of k-anonymity.

Briefly, our approach works in four steps:

1. Clients that are to participate in the FL task download from the server the model with initialized weights.
2. The privacy module of each client decides what data can be shared with the server. Thus, the client can start the model training with its local data. Similarly, the server can also train the model with the data received from the clients.
3. The model updates coming from both the clients and the server are collected and aggregated to improve the global model.
4. The updated model is passed from the server to the clients, and a new iteration loop begins to improve the model.

To show the effectiveness of our proposal, we describe how k-anonymity and l-diversity approaches are integrated into our hybrid federated learning system, and we describe the advantages and disadvantages of our proposal.

The rest of the paper is organized as follows. In Section 2, we introduce the concept of federated learning. Section 3 discusses related work. In Section 4, we present our proposed hybrid architecture focusing on the privacy

[1]Quasi-identifiers are attributes that identify individuals using external sources, [35].

module. Section 5 discusses how our approach can be implemented in a specific scenario. Section 6 provides a discussion on the validity of the proposal and discusses future directions.

2. Preliminaries

In this section, we provide the background needed to understand the proposal, which focuses on federated learning.

Conventional machine learning algorithms are performed in a centralized data center wherein information proprietors add their data. However, data are privacy sensitive, and data owners are not always willing to distribute. Similarly, this kind of training procedure suffers from data privacy leakage threat. To cope with such privacy challenges, a collaboratively distributed deep learning paradigm, called federated deep learning, has become proposed for distributed devices to train a global model while maintaining the training records at the device side without sharing raw training records [19]. Data are allotted and scattered among distinct clients, and no single node stores the entire data set. The workflow of federated learning is that every client trains a local machine learning model with the usage of its information and uploads it to the centralized version for summarizing and averaging. Then, a global model is performed inside the centralized server. Accordingly, federated learning prevents a single node from failure effectively. Federated learning is much like the traditional distributed machine learning [16]. However, assumptions on local datasets are unique. Particularly, the conventional distributed learning pursuits at optimizing the parallel computing energy, however, data are independent and identically distributed (IID) amongst dis-

tinctive parties. In contrast, federated learning specializes in heterogeneous local datasets, which means that training data can be allotted, non-IID, and unbalanced among numerous clients [38]. That is, every client member trains the identical model along with its local records set. The aim is to obtain a global version with the minimized averaged sum of loss functions amongst all contributors.

Federated learning approach consists of three main phases: initialization, aggregation, and update step. In the initialization step, the central server broadcasts the current model w_t to each client. Following that, each participant k using a fixed learning rate η computes the average gradient g_k for E epochs on its local data set n_k at the current model w_t

$$w_{t+1}^k \leftarrow w_t - \eta g_k$$

In the aggregation step, the central server performs a collection of local gradients. Finally, in the update step, the central server uses the federated averaging algorithm [23] to obtain a new global model w_{t+1} for the next iteration applying the update.

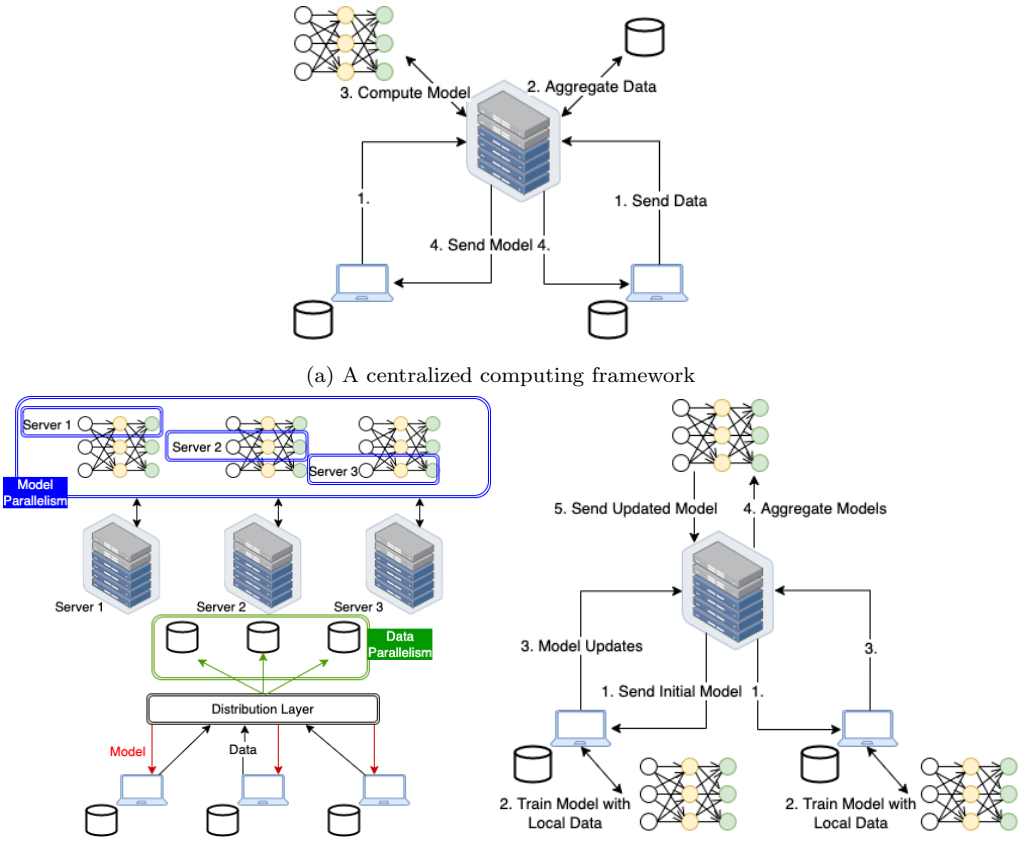
$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{1}$$

where n denotes the global data set. The central server and the clients iterate the above procedure until the global model reaches convergence. This paradigm substantially reduces the risks of privacy leakage since there is no need to directly get access to the training data on client nodes.

3. Related Work

Inspired by the aid of the federated framework, multiple client devices can contribute to the global model training while retaining the original data locally. Applying the Federated Averaging algorithm (FedAvg) [23], in each federated learning round, each contributing device (also known as client), gets an initial model from the main server, runs stochastic gradient descent (SGD) on its local data, and sends back the updated gradients. The server then gathers all gradients from the contributing clients and updates the initial model. This technology can shield the privacy of the collaborating clients successfully since those clients simply need to upload the model gradients or weights derived from their local records instead of their data [20].

Drawing parallels between the federated learning framework and more conventional computing architectures, like centralized and distributed ones, is depicted in [15], where authors discuss recent advances and present an extensive collection of open problems and challenges. In a centralized framework (Figure 1a), all computations are executed on the central server, while the clients request the server to use the computed model. During the training phase, the server needs to have access to the entire dataset constituted by all local sets. Differently, in the distributed computing framework (Figure 1b) there is the presence of multiple servers, also indicated as cloud, dividing up the computational load and data sets that might be anonymized. Finally, as already mentioned, in federated learning (Figure 1c), clients become active contributors to the computational load and only partially share information with the central server. The server acquires a new role: it aggregates the updates from several clients and then broadcasts back the aggregated infor-



(a) A centralized computing framework

(b) A distributed computing framework

(c) A federated learning framework

Figure 1: An illustration of the distinct computing frameworks.

mation to the end clients.

Moreover, federated learning has lately captivated considerable attention, and one of the most relevant research challenges is privacy protection. Liu et al. [21] tackle the privacy issue during model updates. They had the intuition to use sketches commonly employed for network measurement and databases applications. Subject to the model's size, the computed local updates can be a vector of up to millions of numbers. Considering that sketches obfuscate

the original data via independent hash functions, they exploit hash functions to hide the identities of each round of model updates with client owned private hash indices and seeds. Using three representative learning models, their proposed sketching-based federated learning can provide strong privacy guarantees without sacrificing accuracy.

Hao et al. [11] suggested a privacy-enhanced federated learning (PEFL) scheme that is non-interactive in each aggregation. The homomorphic cipher text of private gradients is embedded into augmented learning with error term to achieve their secure aggregation protocol. Their scheme helps prevent private data from being leaked. Nasr et al. [26] presented an exhaustive privacy analysis framework of deep neural networks using white-box membership inference attacks. Attackers can both passively inspect the model updates and actively impact the target model to retrieve additional information. Furthermore, attackers can be equipped with different types of prior knowledge. Authors show that in federated learning, both the server and the clients can perform alarmingly accurate membership inference attacks against other clients. Differently, Wang et al. [37] advanced an architecture incorporating Generative adversarial networks (GANs). Although GANs can reconstruct class representatives of the global data distribution, it is challenging to attack a single client. Authors proposed a practical reconstruction attack named mGAN-AI, which employs GAN with a multitask discriminator, discriminating contemporaneously category and client identity of input data. This facilitates the generator on the malicious server to target and compromise the victim's training samples.

Few kinds of research have combined federated learning and blockchain

as a protection mechanism to defend the privacy of model parameters. Lu et al. [22] present a blockchain-empowered collaborative framework employing differential privacy (DP) and federated learning to protect data privacy further. In addition, data owners can audit the access to shared information. Zhao et al. [40] employ multiple existing technologies to build their system. As usual in a federated learning architecture, clients download and train the global model with local data. Then, the model is sent and stored into the blockchain. In the event of malicious clients or server, the chain is used instead of the central aggregator. Records on the chain are immutable, and any activity is traceable. Both in the work conducted by Lu and in the study by Zhao, the addition of DP noise directly into the local and raw data instead of the gradients can be a drawback of these approaches with a risky incidence on the accuracy.

Orthogonal to all presented strategies, we propose a new hybrid approach that fuses together classic centralized computing framework for machine learning with the new federated learning framework. To the best of our knowledge, this is the first attempt in the scientific community to create a hybrid architecture. With respect to the state of the art, our architecture improves the accuracy of the trained model for federated learning. It reduces the computational load of clients, yet preserving the privacy of their data.

4. Our Proposal

In this section, we describe the architecture of our proposal and its core component, which is the *Privacy module*.

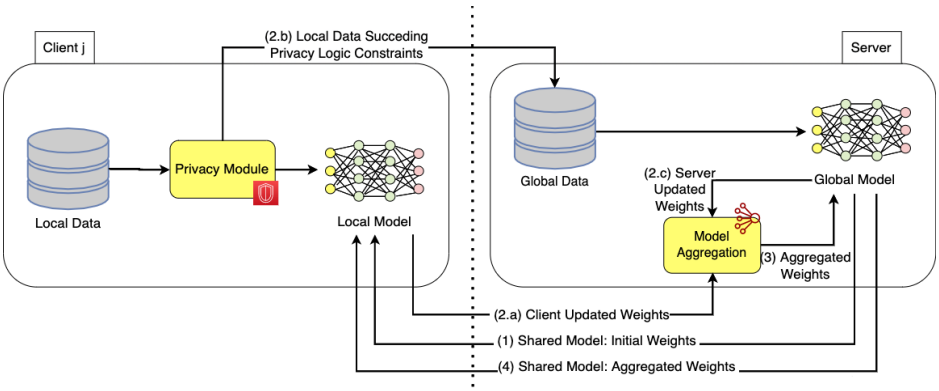


Figure 2: The proposed framework

4.1. Architecture

The basic idea of our proposed hybrid federated learning framework is straightforward: the hybrid architecture can train a unified model across multiple clients while letting the single client decide which data can be kept private and which can be shared with the server. As depicted in Figure 2, our framework proceeds in four steps.

Step 1 (Sharing the unified model with clients): Clients that want to participate in the federated learning task download the model with the initialized weights from the server.

Step 2 (Clients local data extraction and weights update): Each client collects its local data at regular intervals. The privacy module on the client side determines which data can be shared with the server according to the selected privacy logic. Thus, the client can start the model training with its local data (step 2.a). At the same time, the privacy module sends the local data that meets the requirements of the privacy logic to the server (step 2.b). The server trains the model using this data and then sends the

updated weights to the Model Aggregation (step 2.c). Another significant difference with federated learning is that the server can also train its model using data received from the clients' privacy modules. In conclusion, the updated weights are sent from both the clients (step 2.a) and the server (step 2.c) to the Model Aggregation module on the server side.

Step 3 (Updated weights aggregation): The updated weights both originated from the clients and from the server are aggregated/averaged by the Model Aggregation Module in order to enhance the unified model in a collaborative manner. The aggregation approach used in this task is the same as FedAvg (see Equation 1 in Section 2).

Step 4 (Sharing the updated unified model with clients): In this final step, the updated model is passed from the server to the clients, and a new iteration loop to improve the model can begin.

We conclude this section by observing that our framework can be enriched by including other orthogonal features such as a more complex weight updating [36] or applying adaptive optimizers [29]. However, this is out of the scope of our paper.

4.2. Privacy module

In this section, we describe how the privacy module works to decide whether a sample does not violate the user's privacy and, thus, can be forwarded to the server for training.

A sample may contain confidential data if they are classified as sensitive information (e.g., salary, health status) or non-confidential information if they can be released without violating the individual's privacy.

Note that in addition to *identifier* attributes, which allow identification

of the individual, there are also *quasi-identifiers* attributes [35] that, in conjunction with other external information, allow linking and re-identification of an individual. At first glance, removing all identifying attributes might be an option to achieve anonymity [31]. However, thanks to external and publicly available information, the combination of *quasi-identifier* data may lead to re-identification of the data subjects.

Some privacy models can be designed to provide some degree of anonymity. Consider a table $T(a_1, ..., a_n)$ where a_i with $1 \leq i \leq n$ is an attribute.

The concept of k-anonymity is based on the fact that each tuple in the table is indistinguishable from at least $k-1$ other tuples within the table. In other words, the table T satisfies k-anonymity if each tuple of quasi-identifier attributes of the table appears at least with k occurrences (the interested reader can find the formal definition of k-anonymity in [1]). The k-anonymity technique can be achieved through two complementary operations: 1) generalization, which replaces a quasi-identifier value with a general value and, 2) suppression, which consists of removing some values of quasi-identifiers (e.g., a single cell, column, tuple).

Another approach to obtaining privacy is l-diversity. Consider equivalence classes $E_i, ..., E_m$ consisting of tuples t that share a combination of sensitive attribute values. A table guarantees l-diversity if, for each $E_i, ..., E_m$, there exists at least l well-represented values of each sensitive attribute (the interested reader can find the formal definition of l-diversity in [1]).

We conclude this section by formalizing the algorithm used to implement the privacy module. We recall the standard notation from machine learning: Let $D = \{s_1, \ldots, s_{|D|}\}$ be a data set of samples, F be the descriptive

attributes of samples, and $l(s_i)$ (label) be the class to which the sample s_i belongs.

Let $I = \{i_1, \ldots, i_{|I|}\} \subseteq F$ be a subset of features, which is an input of the algorithm: how to determine such a set depends on the specific application (this aspect is discussed in Section 5). Given a sample s_a and a feature i_b, we denote by $\text{val}(s_a, i_b)$ the value of the sample for the feature i_b.

A client runs Algorithm 1 to decide whether a sample can be sent to the server or not. In this algorithm, the input is the set of all samples, and the output is a vector that stores the value *false* at position x if the x-th sample can be sent to the server without violating privacy requirements. A key-value mapping associates a set of attribute values (keys) with a non-negative integer (values) that counts how often such values occur in the dataset. With a little abuse of notation for simplifying the presentation, Line 1 represents the initialization of such counters to zero. Then, the mapping is updated by incrementing the value of each key by one each time a sample with such a key is found (Lines 2-5). Once the cardinality of each key is calculated, in the final step (Lines 6-11), the x-th sample is marked as to be sent to the server (Line 9) if its key is present in at least k other samples.

For the sake of completeness, we specify in Algorithm 2 the procedure carried out by each client when l-diversity is applied. In this case, the input also contains the labels of the samples, and the main difference from Algorithm 1 is that the key includes the label value in such a way that there are at least l samples with the same values of both the features in I and the label.

It is worth noting that the proposed algorithms also work with features

Algorithm 1: k-anonymity

Input k: privacy protection degree

Input D: samples

Output $keep$: boolean vector

1: $M[*] = 0$
2: **for each** $s_x \in D$ **do**
3: $\quad key = \Big(\text{val}(s_x, i_0), \ldots, \text{val}(s_x, i_{|I|})\Big)$
4: $\quad M[key] + +$
5: **end for**
6: **for each** $s_x \in D$ **do**
7: $\quad key = \Big(\text{val}(s_i, i_0), \ldots, \text{val}(s_i, i_{|I|})\Big)$
8: $\quad$ **if** $M[key] \geq k$ **then**
9: $\quad\quad keep[x] =$ **false**
10: $\quad$ **end if**
11: **end for**
12: **return** keep;

that have an unbounded number of combinations. For example, consider non-categorical features such as <name, surname, city> and assume that the desired privacy protection degree is $k = 10$. In this case, if there are at least k people with the same name, surname, and city, we can disclose some information related to them. For example, we could disclose that "Giuseppe Rossi living in Rome is COVID positive" without violating the user's privacy (according to the privacy protection degree set above) because there are about 28 citizens with this name/surname in Rome (the name Giuseppe and

Algorithm 2: l-diversity

Input l: privacy protection degree

Input D: samples

Input L: sample labels

Output *keep*: boolean vector

1: $M[*] = 0$
2: **for each** $s_x \in D$ **do**
3: $\quad key = \Big(\text{val}(s_x, i_0), \ldots, \text{val}(s_x, i_{|I|}, l(s_x))\Big)$
4: $\quad M[key] + +$
5: **end for**
6: **for each** $s_x \in D$ **do**
7: $\quad key = \Big(\text{val}(s_i, i_0), \ldots, \text{val}(s_i, i_{|I|}, l(s_x))\Big)$
8: $\quad$ **if** $M[key] \geq k$ **then**
9: $\quad\quad keep[x] =$ **false**
10: $\quad$ **end if**
11: **end for**
12: **return** keep;

the surname Rossi are very common in Italy).

Moreover, observe that k-anonymity and l-diversity also works for non-tabular data, such as images. It is sufficient to define a measure of similarity between two images so that we can determine if two images have the same "key" (this is used in Lines 3 and 4 of Algorithms 1 and 2). The reader interested in this aspect can find an example of using k-anonymity for images in [25].

A final remark is that the choice of the approach to be used in the privacy module and the value of k or l are parameters of our architecture and strongly depend on both the specific application context and the required privacy protection degree. An example of application of these approaches is presented in the next section.

5. Case study

This section aims to show how to deploy our approach, particularly the privacy module, in an application scenario. We recall that we propose a new strategy to perform machine learning by a hybrid approach between centralized and federated learning. Thus, the situations in which this approach can be used are extensive: the most crucial aspect is to build the privacy model to handle quasi-identifier attributes and the desired level of privacy protection.

Among all the possible scenarios, probably the most significant and timely one is the case in which we need to build a model to predict COVID-19 severity and we can conduct a study using data of patients with COVID-19 infection hospitalized in a large number of health centers [8, 10]. Federated learning leads each center to participate in the training of a global model while retaining the original data locally.

Unfortunately, just for privacy reasons, there is no dataset available for the problem of predicting COVID-19 severity. The desired dataset should have the following properties:

- to have a large number of samples. Indeed, the dataset should provide data for both training and testing sufficient for any client;

- to have features involved in privacy concerns. This is very important to motivate the use of the privacy module;
- to have been used in other research activities in order to be relevant and sound.

We observe that there are very few datasets offering these three properties. The dataset we use in this section has these properties and refers to a prediction problem that is general and exhaustive. Specifically, we consider a classification task to predict whether a person's income exceeds a given threshold. Knowing some details about this person, in some situations, could allow the person identification. The dataset is known by the name *Adult Census Income* and has been widely analyzed in many other pieces of research [18, 13, 4, 30]. These data are extracted from the Census bureau database [34] and contain 14 features, including *age, race, sex, native country, education, occupation, marital status, relationship, capital gain or loss,* and *working hours per week*. The attribute *income* is the attribute to predict and states whether a person makes over $50K a year (the reader can see [34] for details). The dataset is composed of 32561 rows. In the following, we show how our approach can be instantiated in this scenario, and we compare its performance with respect to both centralized learning and federated learning.

5.1. Pre-processing and setting

To implement our approach, we built a Python simulator that uses the Tensorflow library as learning framework. In the pre-processing task, we follow the same operation as done in [32, 17]: we deleted from the dataset the

2399 rows with missing values, we removed the attribute *fnlwgt* (final weight) because not useful for prediction and the attribute *education* because it is derived from *education-num*. The continuous attributes *hours-per-week*, *capital-gain* and *capital-loss* were mapped to ordinal attributes. The text attributes *workclass, marital-status, occupation, relationship, race, sex, native-country,* and *income* were encoded with integer values between 0 and the number of different classes. We call this first encoding `E1`, and the resulting dataset is available as research data for this paper. Finally, each feature value was scaled to the range [0,1].

Concerning the text attributes *workclass, marital-status, occupation, relationship, race, native-country*, a concern is about the use of *label encoding* (i.e., converting each value of a feature to a number) for nominal (i.e., non-ordinal) features. Thus, we considered a second encoding `E2` that encodes these text attributes by adding a new binary feature for each category, in which the value 1 represents the presence of that category (this is called *One Hot Encoding*).

Concerning the setting of the learning parameters, we used standard choices: we set the training and test sets as 70% and 30% of the whole dataset, respectively, and we used the *accuracy* for evaluating the model performance (as done in [18]).

We used ten clients that train their local model in the considered scenario by the training set. To implement the heterogeneity of the data distribution among the parties [17], we trained each client by samples of people with the same age interval.

5.2. Privacy module

As discussed in section 4.2, the implementation of the privacy module and the value of its parameters strongly depend on the considered application. We implemented both the k-anonymity approach and l-diversity approach, where k and l are parameters whose values depend on the level of privacy required in the considered scenario [3]. When the k-anonymity is used, quasi-identifier attributes have to be detected. We selected *age, race, sex* in our application as quasi identifier attributes. Thus, a client does not process a sample locally if there are at least k people with equal values of the attributes *age, race, sex*. Concerning the parameter k, we used several values in the range $1 \leq k \leq 550$. The privacy module has also been implemented with the l-diversity approach. In this case, a client does not process a sample locally if there are at least l well-represented values for the attribute *income* (i.e., at least l people with equal values of quasi-identifier attributes and *income*).

5.3. Building the model

We need a classification model to solve the *Adult Census Income* classification problem. As usual in this context, we used a neural network. Among the several learning models with different parameters, we used the model proposed in [18]: it uses one hidden layer having the same size as the input layer, ReLU activation, and the Adam optimization as a stochastic gradient descent method.

The accuracy of the selected model measured for the training and test set for each encoding `E1` and `E2` varying the number of epochs is reported in Figure 3. We observe that the second encoding (`E2`) shows lower accuracy, even though One Hot Encoding is the preferred encoding for nominal data.

This result can be explained by considering that nominal encoding may not be as effective when:

1. a large number of categories are present in data. In this case, several dummy features similar to the number of categories have to be added. For example, in our case, encoding `E1` produces 12 features, whereas encoding `E2` generates 95 features. Thus, encoding `E2` introduces sparsity in the dataset because many columns have 0s and a few have 1s. The result is to enlarge the dataset without adding much information.
2. encoded attributes have a low correlation with the class to predict. Thus they do not give a great contribution to the prediction. This is true in our case: we observed that the text attributes considered above have a low correlation with the label income.

Concerning encoding `E1`, which is the one used in the following, we observe that the obtained results are the same as the results reported in Figure 3.(a) of [18], which used the same dataset. This makes us confident that the used model and its implementation are sound.

The results of encoding `E2` show that 1) accuracy on the test set is comparable with that on the training test and 2) a good accuracy (about 0.80) is reached after 100 epochs, and, then, accuracy very slightly increases until 400 epochs (about 0.02), and no further increase is measured after 400 epochs. The latter result allows us to focus on the first 100 epochs.

5.4. Partially-federated learning

In this section, we study how the proposed approach is implemented in the considered scenario.

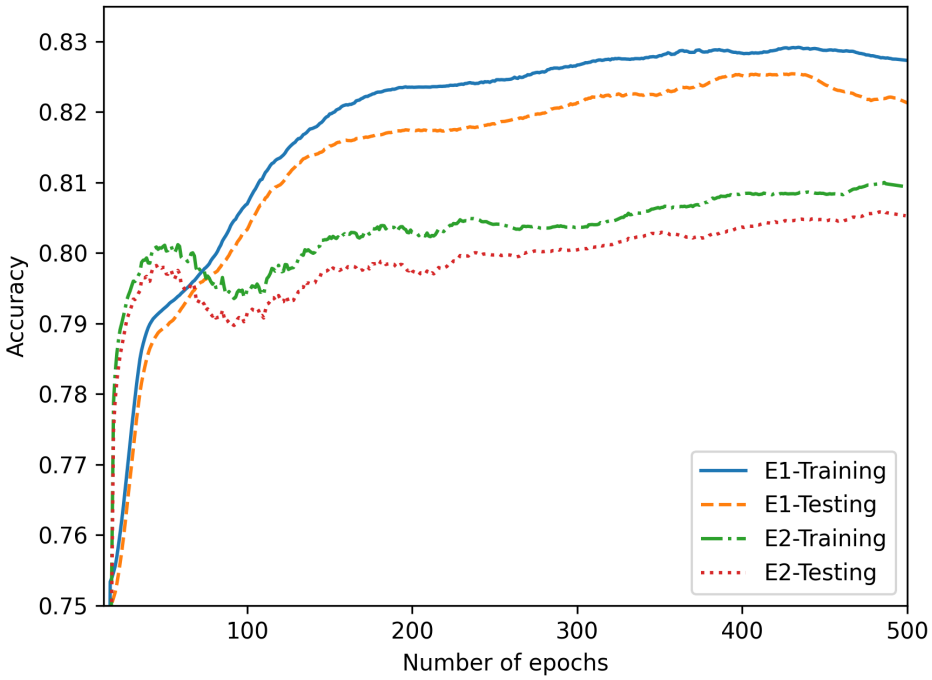


Figure 3: Accuracy of the learning model.

We recall that the privacy module decides whether a sample has to be trained locally or externally (i.e., by the central model) depending on the requested privacy protection degree (see section 4.2). We implemented both the k-anonymity and l-diversity approach and measured the number of samples that are processed externally for different values of k and l. The results of this experiment are reported in Figure 4. We observe that $k = 1$ and $l = 1$ are the lowest privacy-request levels and assume that no privacy issue occurs: thus, all samples can be trained externally, and a client does not have any data to process (the load reduction of each client is 100%). In practice, this

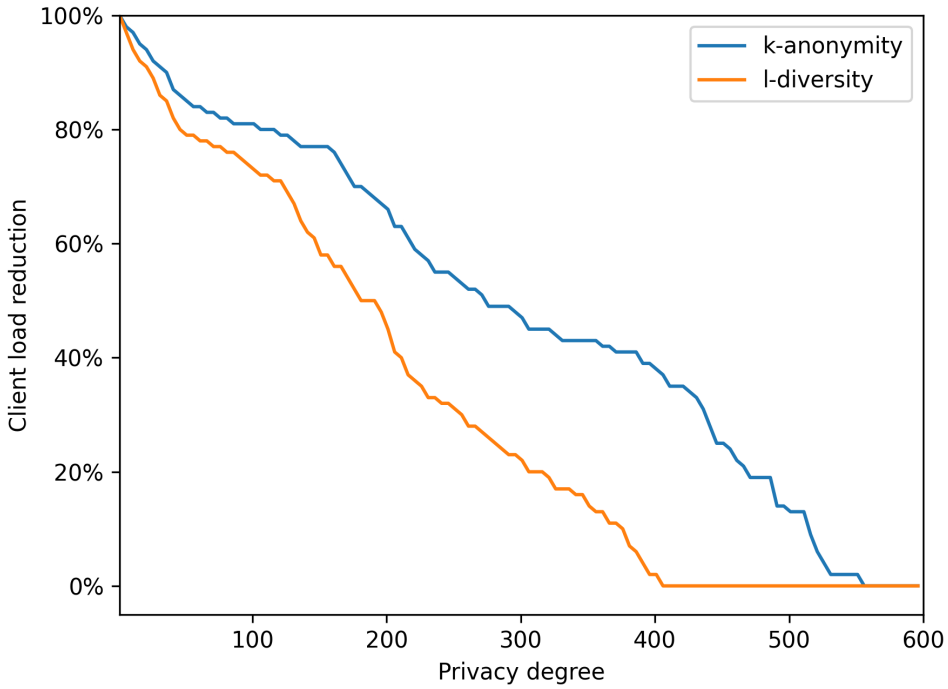


Figure 4: Reduction of the client load vs privacy protection degree k and l.

is a situation in which there is no privacy request, and all data are trained centrally by a server (we denote this case as *C.L.* in the following). In this case, we measure the highest accuracy.

As k and l increase, data are gradually processed by clients, and accuracy decreases, showing how the federated model can be biased towards different clients, as also observed in [24]. Specifically, when $k \geq 554$ or $l \geq 402$ (observe that such thresholds are specific for this dataset), no sample can be sent to the central model because this would result in a privacy violation: as a consequence, the load reduction is zero. This is the case of standard

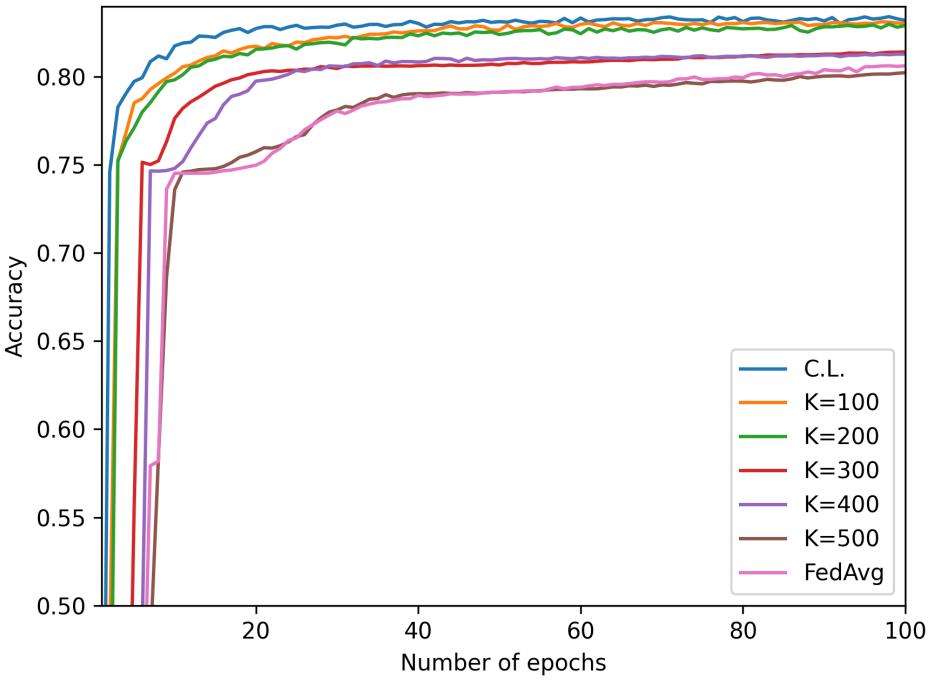


Figure 5: Accuracy vs privacy protection degree (k-anonymity).

federated learning (i.e., *FedAvg*). Intermediate values of k and l produce intermediate load reductions as reported in Figure 4.

This experiment shows the first advantage f our approach, which is to reduce the resources consumption on the client side (i.e., computation power required by the complex task of training the local model). The second advantage concerns the improvement of the accuracy of the model, and this aspect is analyzed in the next experiments.

The third experiment aims to measure the model's accuracy versus the chosen privacy protection degree when the privacy module implements the

k-anonymity approach. The obtained results are depicted in Figure 5. According to the results discussed in the previous experiment, the curve $k = 554$ represents the case of federated learning *FedAvg*, because all samples are processed locally by each client. Thus, this curve has the same trend as the curve `Testing` depicted in Figure 3 (note that the considered number of epochs is different). In contrast, the curve $k = 1$ represents the case of centralized learning *C.L.* (i.e., all samples are trained only by the central model) and shows (i) a quick convergence to the highest accuracy and (ii) the best accuracy. The curves for $1 < k < 554$ show intermediate accuracy and this allows us to conclude that by partially relaxing the privacy constraints we can obtain better and faster converging accuracy than using federated learning. Moreover, by reducing k, the model reaches higher accuracy with fewer training samples (i.e., few epochs are necessary). This experiment highlights the effectiveness of the privacy module in reaching a better classification accuracy after a fixed number of epochs.

In the fourth experiment, we replace the k-anonymity approach with the l-diversity in the privacy module and measure the resulting accuracy again: the result of this experiment is reported in Figure 6. Whereas the curves centralized learning *C.L.* ($l = 1$) and *FedAvg* ($l = 402$) follow the expected trend and we can repeat the same considerations done for the cases $k = 1$ and $k = 554$ of the third experiment, the remaining curves show a surprising trend. In the curves $l = 100$ and $l = 200$, the accuracy decreases after about epoch 20, and the curve $l = 300$ shows less accuracy than $l = 402$: these surprising results lead us to investigate the matter further.

Consequently, we measure the accuracy after 100 epochs for $1 \leq l \leq 402$

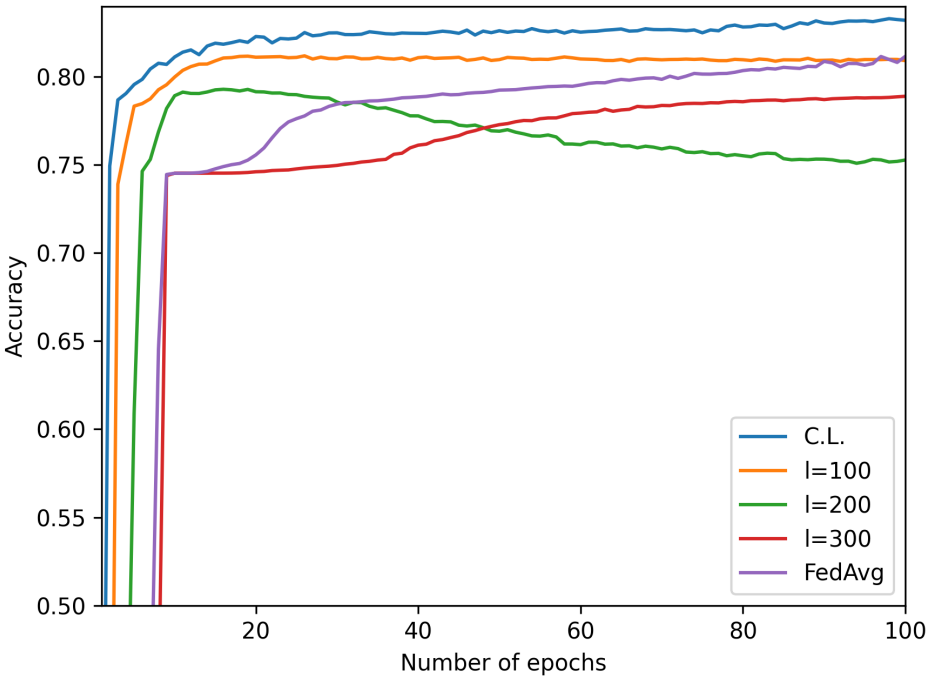


Figure 6: Accuracy vs privacy protection degree (l-diversity).

with higher granularity, and the measured values are reported in Figure 7. We observe that the l-diversity approach allows us to improve the accuracy of the model if $l \leq 200$, whereas higher values of l do not improve the model performance.

However, the most evident result is that the obtained curve is very different from that of the k-anonymous approach, and we further investigate to understand why. Finally, to provide an intuitive explanation, we analyze the loss of accuracy and find two main reasons. The first reason for accuracy loss is because the federated learning approach (which occurs when $l = 402$)

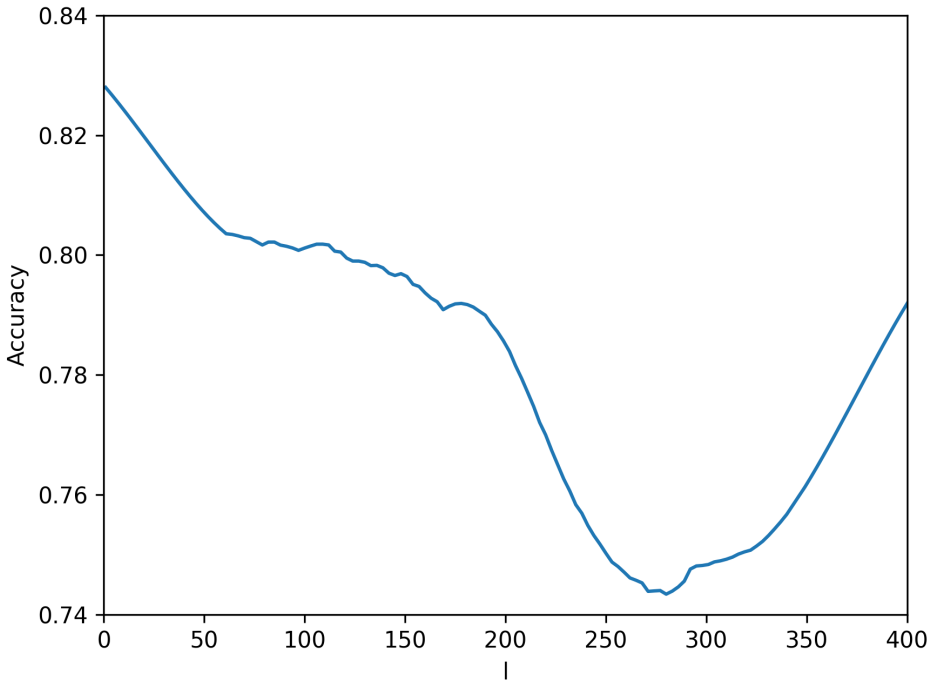


Figure 7: Accuracy vs privacy protection degree (l-diversity).

results in reduced accuracy with respect to the centralized learning (when $l = 1$). We can try to measure this error by exploiting the results of the fourth experiment, where the accuracy reduction was measured when the k-anonymity approach was implemented. We report this error in Figure 8 by the curve named 'component 1'.

Implementing the l-diversity approach introduces the second component of error that is explained by recalling how l-diversity works. Specifically, a sample with the attribute' income' higher than or equal to 50k is sent to the server if there are at least l other samples with the same characteristics. The

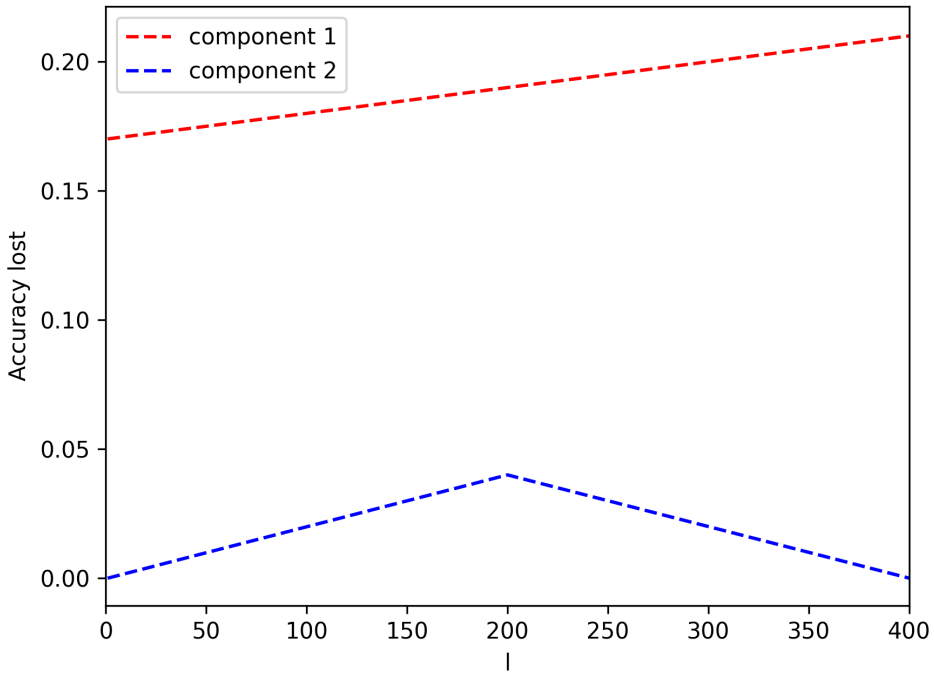


Figure 8: Accuracy loss (l-diversity).

same filter is applied to samples lower than 50k. However, by looking at the dataset, we realize that the two classes of the attribute *income* (i.e., '$\geq$ 50k' and '$<$ 50k') have different frequencies. Thus, for l increasing, the server receives more samples of a class, and the clients receive more samples of the other class. The consequence is that both models (i.e., the local models of the client and the global model of the server) are biased toward one or the other class and the accuracy is reduced. This phenomenon reaches the maximum effect when the difference in the number of samples trained by each model is minimum. By looking at Figure 4, we observe that for l about 200, 50%

samples are sent to the server, and 50% are trained by clients. Thus, this is the point at which the considered component of the error is maximum. We report in the curve 'component 2' of Figure 7 the accuracy trend when only this component is present. However, observe that the curves' component 1' and 'component 2' represent just ideal trends because their real values cannot be measured. This is why the reader cannot find a complete correspondence between the curves illustrated in Figures 7 and 8.

In the last experiment, we compare the running time of the centralized approach (i.e., $k = l$ or $l = 1$), federated learning *FedAvg* (i.e., $k \geq 554$ or $l \geq 402$), and our hybrid partially-federated learning technique (i.e., remaining values of k and l). The aim of this experiment is to determine how the additional tasks required to execute our approach (i.e., determining whether a sample is sensitive or not and sending that sample to the server) affect the overall task.

As in the previous experiments, the scenario under consideration has ten clients and one server, and we measure the time required to execute 100 epochs. In this experiment, we need to set two additional parameters related to communication rate and computing capability. We set a transmission rate of 500 KB/s for communication between clients and server, which is a low rate for current technology. As for the computing capability, we distinguish two cases: 1) equality of computational power between server and clients, and 2) inequality of computational power, where the server is ten times faster than the clients.

In Table 1 and Table 2 we show the results of this last experiment. We measure the following values:

- the time taken by the server to build the model (column *Server side*);
- the time taken by the slowest client to train the model (column *Client side*), since the clients are in parallel;
- the time taken to transfer data from the clients to the server (column *Transfer*) also in this case, transfers are in parallel);
- the total running time as the sum on the individual times (column *Total*).

Observe that the time taken to transfer model weights between the server and the clients is negligible with respect to data size.

Table 1 shows the results for the first case where the computational power of the server and clients is equal. We analyze these results starting from the case of centralized learning (i.e., $k = 1$ or $l = 1$). All clients run the privacy module and send their data to the server in about 0.79s, whereas the server takes about 126s to train the model. For increasing values of k and l, a smaller amount of data are sent to the server, and the parallel processing of clients can reduce the execution time. The last row shows that when *FedAvg* is used, no data are transmitted (*transfer* is zero) and, thus, the server processing time is zero.

The results for the second case, where the server's computing power is 10 times that of a client, are shown in Table 2. This value was chosen to "cancel" the advantage of distributed computing power given by the federated approach (indeed, there are 10 clients). We find that the best execution time (about 13s) holds for centralized learning ($k = 1$ and $l = 1$): also, in

k	Server side	Client side	Transfer	Total
C.L.	126,00	0,00	0,79	126,79
100	110,64	13,18	0,69	124,51
200	78,58	15,05	0,59	94,22
300	66,97	15,95	0,41	83,34
400	42,53	19,55	0,27	62,36
500	24,12	21,13	0,12	45,38
FedAvg	0,00	24,75	0,00	24,75

l	Server side	Client side	Transfer	Total
C.L.	126,00	0,00	0,79	126,79
100	91,88	5,61	0,63	98,12
200	55,52	13,28	0,41	69,21
300	31,71	22,41	0,35	54,47
FedAvg	0,00	24,75	0,00	24,75

Table 1: Running time in seconds with equal computational power of server and clients.

this case, the time required to transfer data (about 0.79s) is negligible with respect to the total time to build the model.

In summary, the analysis of the two results shows that most of the time is spent on the model training, whereas the data transfer has a negligible impact on the total running time.

k	Server side	Client side	Transfer	Total
C.L.	12,60	0,00	0,79	13,39
100	11,06	13,18	0,69	24,93
200	7,86	15,05	0,59	23,50
300	6,70	15,95	0,41	23,06
400	4,25	19,55	0,27	24,08
500	2,41	21,13	0,12	23,67
FedAvg	0,00	24,75	0,00	24,75

l	Server side	Client side	Transfer	Total
C.L.	12,60	0,00	0,79	13,39
100	9,19	5,61	0,63	15,43
200	5,55	13,28	0,41	19,24
300	3,17	22,41	0,35	25,93
FedAvg	0,00	24,75	0,00	24,75

Table 2: Running time in seconds in the case of inequality of computational power, with the server 10 times faster than each client.

6. Discussion and Conclusion

In this section, we discuss the advantages and disadvantages of the proposed approach. In the literature, federated learning approaches assume that each client trains all data locally for privacy reasons. However, there are many application contexts where some data should be processed locally by a client. In contrast, other data can be sent to a server without causing a privacy problem: an example was presented in Section 5. Therein, if there

are too few participants (i.e., below a certain threshold) with equal values of quasi-identifier attributes, their personal data will be processed locally by the client, otherwise, their data will be sent to the server for training the model. Of course, this possibility should be included in the privacy policy.

For construction, our approach gives results similar to centralized learning when privacy protection degree is low and similar to federated learning when privacy protection degree is high. Incidentally, our approach includes centralized and federated learning as special cases. It is well known that centralized learning is usually more performing than federated learning. Thus, the possibility of reducing the privacy protection degree provided by the proposed approach can improve (in efficacy and efficiency) the prediction task with respect to federated learning.

Observe that, we do not compare the proposed architecture with other proposals different from standard federated learning (*FedAvg*) and centralized learning since our proposal is orthogonal to these studies. In principle, we could choose an improved version of federated learning and modify it by applying our strategy to allow some data to be processed centrally. Whenever centralized learning is more accurate than federated learning (this happens often), we improve the accuracy of such a particular federated learning technique.

Another observation is about the use of accuracy to measure the performance of the prediction task. Accuracy is one of the most commonly used metrics in this domain: for example, accuracy was used in [18] for the same dataset. Of course, we could also use other metrics, such as *sensitivity* when minimizing false negatives or *specificity* when minimizing false

positives. However, in the scenario considered here, there is no such need. Even though accuracy is not the best metric to measure the performance of a model in absolute terms, we can use it since we performed a comparative analysis between centralized learning, partially federated learning (i.e., our approach), and federated learning. From a quantitative point of view, using k-anonymity, we can achieve up to 3.75% improvement in accuracy (from 0.80 to 0.83) by reducing k and achieving faster convergence to maximum accuracy (i.e., fewer epochs are needed to achieve the same accuracy). Concerning the use of l-diversity, we have similar results when classes are balanced, while in the case of unbalanced classes, the approach is only effective for low values of l (l about less than half the maximum value).

Besides improving performance, another interesting result is that using k-anonymity or l-diversity reduces the load of client computations when k and l decrease: this is an essential benefit since client hardware and computational resources are typically more limited than those of a server (e.g., a client may not be able to perform training of extensive learning models). The use of the privacy module with appropriately set parameters k or l (their value depends on the particular scenario) allows us to reduce the amount of data to be trained locally without violating privacy.

In summary, our architecture is a hybrid solution between the centralized and federated learning systems. By appropriately setting the privacy level (i.e., the values of k and l), we can move towards one or the other and thus achieve benefits in terms of client load and model accuracy without violating privacy constraints.

This hybrid architecture can train a unified model across multiple clients

while letting each client decide which data must be kept private and which data can be shared with the server. Specifically, our system enforces both the k-anonymity and l-diversity model to protect the privacy of clients' data.

To the best of our knowledge, this is the first attempt in the scientific community to create such a hybrid architecture.

As future steps, we intend to study how the proposed framework can be enriched by including other orthogonal features such as a more complex weight updating [36] or applying an adaptive optimizer [29].

References

[1] C. C. Aggarwal and S. Y. Philip. *Privacy-preserving data mining: models and algorithms*. Springer Science & Business Media, 2008.

[2] G. Andrew, O. Thakkar, H. B. McMahan, and S. Ramaswamy. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

[3] F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera. A privacy-preserving localization service for assisted living facilities. *IEEE Transactions on Services Computing*, 13(1):16–29, 2016.

[4] N. Chakrabarty and S. Biswas. A statistical approach to adult census income level prediction. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 207–212. IEEE, 2018.

[5] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li. A training-integrity

privacy-preserving federated learning scheme with trusted execution environment. *Information Sciences*, 522:69–79, 2020.

[6] Y.-R. Chen, A. Rezapour, and W.-G. Tzeng. Privacy-preserving ridge regression on distributed data. *Information Sciences*, 451:34–49, 2018.

[7] Cisco. Cisco annual internet report (2018–2023) white paper. `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`, 2020.

[8] S. Domínguez-Rodríguez, S. Villaverde, F. J. Sanz-Santaeufemia, C. Grasa, A. Soriano-Arandes, J. Saavedra-Lozano, V. Fumadó, C. Epalza, M. Serna-Pascual, J. A. Alonso-Cadenas, et al. A bayesian model to predict covid-19 severity in children. *The Pediatric Infectious Disease Journal*, 40(8):e287–e293, 2021.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[10] M. Fisichella. Unified approach to retrospective event detection for event- based epidemic intelligence. *Int. J. Digit. Libr.*, 22(4):339–364, 2021.

[11] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Transactions on Industrial Informatics*, 16(10):6532–6542, 2020.

[12] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[13] T. Hu, V. Iosifidis, W. Liao, H. Zhang, M. Y. Yang, E. Ntoutsi, and B. Rosenhahn. Fairnn-conjoint learning of fair representations for fair decisions. In *International Conference on Discovery Science*, pages 581–595. Springer, 2020.

[14] C. Jiang, C. Xu, and Y. Zhang. Pflm: Privacy-preserving federated learning with membership proof. *Information Sciences*, 576:288–311, 2021.

[15] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[16] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[17] Q. Li, Y. Diao, Q. Chen, and B. He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021.

[18] G. Liu, C. Wang, X. Ma, and Y. Yang. Keep your data locally:

Federated-learning-based data privacy preservation in edge computing. *IEEE Network*, 35(2):60–66, 2021.

[19] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, 2021.

[20] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal*, 7(8):7751–7763, 2020.

[21] Z. Liu, T. Li, V. Smith, and V. Sekar. Enhancing the privacy of federated learning with sketching. *CoRR*, abs/1911.01812, 2019.

[22] 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*, 16(6):4177–4186, 2020.

[23] B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282. JMLR: W&CP, 2017.

[24] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic federated learning. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceed-*

ings of Machine Learning Research, pages 4615–4625. PMLR, 09–15 Jun 2019.

[25] V. Mygdalis, A. Tefas, and I. Pitas. Introducing k-anonymity principles to adversarial attacks for privacy protection in image classification problems. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2021.

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

[27] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, and C. S. Hong. A crowdsourcing framework for on-device federated learning. *IEEE Transactions on Wireless Communications*, 19(5):3241–3256, 2020.

[28] E. Parliament and C. of the European Union. General data protection regulation. `https://gdpr-info.eu/`, 2016.

[29] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan. Adaptive federated optimization. *CoRR*, abs/2003.00295, 2020.

[30] V. Ríos Canales. Using a supervised learning model: Two-class boosted decision tree algorithm for income prediction. *Computer Engineering;*, 2016.

[31] J. Salas and J. Domingo-Ferrer. Some basics on privacy techniques, anonymization and their big data challenges. *Mathematics in Computer Science*, 12(3):263–274, 2018.

[32] A. Samal. Package arules. `https://www.coursehero.com/file/26739155/arulespdf`, 2017. Accessed 15-December-2021.

[33] P. Samarati. Protecting respondents identities in microdata release. *IEEE transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[34] UCI Machine Learning. Adult census income. `https://www.kaggle.com/uciml/adult-census-income`, 2021. Accessed 15-December-2021.

[35] S. d. C. d. Vimercati and S. Foresti. Quasi-identifier. *Encyclopedia of Cryptography and Security*, pages 1010–1011, 2011.

[36] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, pages 1–13, 2020.

[37] 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*, pages 2512–2520, 2019.

[38] R. Younis and M. Fisichella. FLY-SMOTE: re-balancing the non-iid iot edge devices data in federated learning system. *IEEE Access*, 10:65092–65102, 2022.

[39] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[40] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu. Privacy-preserving blockchain-based federated learning for iot devices. *IEEE Internet of Things Journal*, 8(3):1817–1829, 2021.