Future Generation Computer Systems 149 (2023) 200–211

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

FedHEONN: Federated and homomorphically encrypted learning method for one-layer neural networks

Oscar Fontenla-Romero [*], Bertha Guijarro-Berdiñas, Elena Hernández-Pereira, Beatriz Pérez-Sánchez

Universidade da Coruña, CITIC, Campus de Elviña s/n, 15071, A Coruña, Spain

A R T I C L E   I N F O

Article history:
Received 31 March 2023
Received in revised form 12 June 2023
Accepted 14 July 2023
Available online 19 July 2023

Keywords:
Federated learning
Neural networks
Homomorphic encryption
Privacy-preserving
Edge computing

A B S T R A C T

Federated learning (FL) is a distributed approach to developing collaborative learning models from decentralized data. This is relevant to many real applications, such as in the field of the Internet of Things, since the models can be used in edge computing devices. FL approaches are motivated by and designed to protect privacy, a highly relevant issue given current data protection regulations. Although FL methods are privacy-preserving by design, recently published papers show that privacy leaks do occur, caused by attacks designed to extract private data from information interchanged during learning. In this work, we present an FL method based on a neural network without hidden layers that incorporates homomorphic encryption (HE) to enhance robustness against the above-mentioned attacks. Unlike traditional FL methods that require multiple rounds of training for convergence, our method obtains the collaborative global model in a single training round, yielding an effective and efficient model that simplifies management of the FL training process. In addition, since our method includes HE, it is also robust against model inversion attacks. In experiments with big data sets and a large number of clients in a federated scenario, we demonstrate that use of HE does not affect the accuracy of the model, whose results are competitive with state-of-the-art machine learning models. We also show that behavior in terms of accuracy is the same for identically and non-identically distributed data scenarios.

1. Introduction

Federated learning (FL) is becoming a very active research area in machine learning (ML) because it enables distributed and collaborative training of models. This key feature of many real-world applications allows models to run on edge computing devices which, in many cases, have very limited computational capabilities. Despite most data processing still taking place in centralized data centers, organizations are beginning to discover the benefits of edge computing. Recent studies have predicted that by 2025, due to the use of mobile devices and the internet of things (IoT), 75% of data will be created and processed outside a traditional data center [1], leading to a rethinking of where computing should take place.

FL, introduced in 2017 by McMahan et al. [2], is an ML environment in which many clients collaboratively train a model under the control of a central server while keeping the training data decentralized. The underlying idea of training models without the raw training data being collected in a single location has proven useful in a wide range of practical scenarios. The algorithm proposed by McMahan et al. has been applied to Google's Gboard [3] to improve next word prediction models, and has also been used in other applications in a wide range of fields, such as mobile devices [4,5], finance [6], industrial engineering [7–9], and healthcare [10–13]. Additionally, several studies have explored the use of FL in scenarios reflecting privacy-sensitive data, such as health diagnoses, collaborations across multiple hospitals or government agencies, etc [14–16]. This paradigm presents several advantages compared to traditional centralized ML, among them (a) highly efficient use of network bandwidth, as less information is transmitted; (b) low latency, as real time decisions can be made locally at the different clients, and (c) privacy, as storing data locally rather than replicating it in the cloud reduces the risk of attacks and (assuming that the clients and servers are non-malicious) enhances privacy.

There is a significant body of related work that seeks to analyze and learn from distributed data without exposing sensitive information. Although privacy-preserving data analyses have been conducted for more than 50 years, it is only in the past decade that solutions have been widely deployed at scale. The issue has inspired significant interest recently, both from the

[*] Corresponding author.
E-mail addresses: oscar.fontenla@udc.es (O. Fontenla-Romero), berta.guijarro@udc.es (B. Guijarro-Berdiñas), elena.hernandez@udc.es (E. Hernández-Pereira), beatriz.perezs@udc.es (B. Pérez-Sánchez).

https://doi.org/10.1016/j.future.2023.07.018

research and applied perspectives [17–21]. The fact that FL algorithms preserve data privacy by design is very relevant given current data protection regulations, as systems can achieve much stronger privacy preservation on the basis that distributed devices do not need to share local data, but can deliver the parameters of a local model trained with local data to a centralized server.

However, recent research has shown that local data of distributed devices can be leaked through the trained local model parameters. The possibility therefore exists that a centralized server or attacker can infer/extract sensitive private information using the structure and parameters of local learning models. Stronger privacy properties are possible when FL is combined with other technologies, with different privacy techniques recently established for FL that are briefly described and compared below.

Differential privacy (DP) [22,23]. DP techniques introduce a level of uncertainty into the ML model that masks the contribution of any individual user. However, when the amount of data is small, introducing noise will inevitably affect model training. Moreover, since the data still has to be transmitted elsewhere, there has to be a trade-off between accuracy and privacy. To add client-side data protection, [24] introduced the DP approach to FL, hiding the client's contributions during training. While DP techniques are computationally efficient, they inevitably degrade the predictive performance of the model.

Secure multi-party computation (SMC) [25]. SMC, which incorporates a family of cryptographic techniques involving multiple parties, provides proofs of security in a well-defined simulation framework that fully guarantees that each party knows nothing except what is contained in its raw training data. While such zero knowledge about individual contributions of other clients is desirable, it usually requires complicated computation protocols and consequently may not be achieved efficiently. In certain scenarios, partial knowledge disclosure may be considered acceptable if security guarantees are provided. Unlike DP, SMC delivers the exact same result as its unencrypted counterpart.

Homomorphic encryption (HE) [26]. HE protects user privacy by applying an encryption mechanism to the model parameters exchanged during learning. The model itself is not transmitted, nor can it be guessed from the other party's data, and so there is little possibility of leakage at the raw data level. While HE approaches sacrifice computational cost (due to encryption) for security, it is still more efficient than SMC and is also typically more accurate than DP because it does not introduce noise into the data.

In this paper, we present FedHEONN, a novel federated ML method based on one-layer neural networks that incorporate HE to ensure robustness against model inversion attacks. The remainder of this paper is structured as follows. Section 2 reviews the main works on FL and HE. Section 3 details the proposed method. Section 4 shows the results of experiments carried out with large data sets in independent and identically distributed (IID) and non-IID scenarios. Finally, Section 5 presents the conclusions of the work.

2. Related work

Some of the main challenges of FL environments are related to the characteristics of the data silos in the clients, and the communications between the clients and/or with the server, specifically: (1) dealing with unreliable and relatively low bandwidth connections, (2) the unavailability for training of many or several clients at certain times, and (3) the great heterogeneity (non-IID) that may exist between data from different clients and/or data imbalances in each client [27].

FedAVG [2] can be considered the de facto standard in the federated optimization setting. Based on iterative averaging, it can be applied to almost any (deep) model that can be trained with stochastic gradient descent (SGD). In this scheme, in each round, a fraction of the clients are selected to individually perform several SGD iterations and to compute a global model update using only their local training data. Subsequently, a server averages the local model updates to compute the new global model. In addition to its satisfactory performance, FedAVG tackles communication bottlenecks by considerably reducing the frequency (rounds) of communications required. This is a desirable condition in FL since, in addition to the fact that communication may be limited, its cost in these environments tends to predominate over the computational cost. However, for unbalanced and/or non-IID data, although FedAVG is robust in certain applications, due to the SGD iterations on each client, in certain circumstances it tends to overfit to local data, resulting in a low convergence rate or instability. This phenomenon is called client-drift [28].

SCAFFOLD has been described as a solution to this problem by Karimireddy et al. [29]. It uses control variates to estimate client-drift with respect to the model in the server, and uses these to correct local updates. As a consequence, it has the advantages that it requires significantly fewer communication rounds, is not affected by data heterogeneity or client sampling, and is at least as fast as SGD. Additionally, it can exploit similarity between clients to further reduce communications and accelerate convergence. However, SCAFFOLD is only designed to work in the cross-silo setting, participated in by only a relatively small number of reliable clients usually having large computational capabilities (e.g. institutions).

MIME [30], in contrast with SCAFFOLD, can adapt an arbitrary centralized algorithm to FL to work in a cross-device setting, where clients may have limited computing resources (e.g. mobiles), the communication network may be unreliable, and the number of clients may be so great that during training, not a single pass may ever be made by all clients. Like SCAFFOLD, MIME uses control variates that, at the client level, are combined with a global state of the optimizer computed at the server that remains fixed during each local learning round. This avoids excessive adaptation to a client by ensuring that each local update mimics that of the centralized method running in an IID scenario. It also reduces communication by naively reducing the number of participating clients per round. As a result, this framework is able to translate the convergence of a centralized algorithm to the federated environment where, under certain circumstances speed may be even greater.

As mentioned, a major challenge in FL is privacy, so the data exchanged during learning between clients must be protected to prevent the inferral of sensitive information. HE refers to encryption schemes that allow certain calculations to be performed directly on encrypted data without first decrypting the data. The results of encrypted computations are identical to those obtained for operations performed on the unencrypted data. Thus, encrypted calculations may only be decrypted with the secret key known exclusively to the data owner.

HE carries a computational overhead, however, as computations already costly for unencrypted data probably become unfeasible for encrypted data. Nonetheless, use of this technology is relevant in scenarios with very strict privacy requirements, where computation in the cloud without encryption is not feasible, and where, because only certain data need to be encrypted, the computational overhead is small.

Multiple HE schemes with different capabilities and trade-offs have been proposed in recent years, ranging from fully HE [31] to more efficient leveled variants [32–35]. To date, however, there has been little research into the HE approach to FL. In [36] a FL scenario is assumed in which the data are vertically divided (by features) among clients and only one client knows the target variable. Their proposed solution, given that there is no linkage between the partially collected entities at each client, is as follows: to learn a linear model by privacy preserving entity resolution and using a Taylor approximation applied to the loss and gradient functions, in order to work over messages encrypted with an additively homomorphic scheme for privacy-preserving computations. However, this method only focuses on binary classification with two clients. In another approach, described in [37], a secure system is proposed to protect logistic regression training data using additive HE; however, the focus is on a distributed scenario, not necessarily federated; in a practical application, certain issues would need to be resolved, such as excessive computational overhead [38]. Proposed more recently in [39] is a solution for distributed deep neural network learning that uses asynchronous SGD in combination with additive HE to preserve privacy. Although accuracy is identical to that of a centralized deep learning system with tolerable communication and computational overheads, this method has not been evaluated to demonstrate that, in contrast with FedAVG, it is not sensitive to the client-drift problem.

3. Proposed method

For one-layer feedforward neural networks (with no hidden layers), we propose a new FL method that uses HE to ensure robustness against model inversion attacks. Despite some limitations, simple models are still of great interest as they can solve many tasks, are easier to interpret, can be trained relatively quickly, and can handle large data sets efficiently. We first define the terminology based on a model with a centralized training set and then describe the FL scheme in detail.

Consider a centralized training data set defined by an input matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$, where m is the number of inputs (including the bias) and n is the number of data instances. The desired output matrix is defined by $\mathbf{d} \in \mathbb{R}^{n\times c}$, where c is the number of outputs. In what follows, we consider only one output (i.e., $\mathbf{d} \in \mathbb{R}^{n\times 1}$) to avoid cumbersome derivation; however, the extension to multiple outputs is straightforward, since each output of the one-layer neural network depends only on a set of independent weights. The neural network parameters are defined by a weight vector (including the bias), $\mathbf{w} \in \mathbb{R}^{m\times 1}$, and, therefore, the output of the model ($\mathbf{y}$) is as follows:

$$\mathbf{y} = f(\mathbf{X}^T\mathbf{w})$$

where $f : \mathbb{R} \rightarrow \mathbb{R}$ is the nonlinear activation function at the output neuron. Our proposal is based, in part, on previous research, but for reasons of clarity and completeness, we describe it briefly in the next subsection.

3.1. Background

In neural networks, optimal weights are usually obtained through an iterative process that minimizes a cost function. In the case of supervised learning, while several alternatives for the cost function are available, the mean squared error (MSE) is one of the most used. Typically, the MSE at the output of the network is measured by comparing the actual output $\mathbf{y}$ and the desired output $\mathbf{d}$. However, as was previously shown [40], an alternative option is to minimize the MSE measured *before* the activation function, i.e., between $\mathbf{X}^T\mathbf{w}$ and $\bar{\mathbf{d}} = f^{-1}(\mathbf{d})$. In addition, to avoid overfitting, a regularization term based on the L2 norm can be incorporated, resulting in the following cost function [41]:

$$J(\mathbf{w}) = \frac{1}{2}\left[\left(\mathbf{F}\left(\bar{\mathbf{d}} - \mathbf{X}^T\mathbf{w}\right)\right)^T\left(\mathbf{F}\left(\bar{\mathbf{d}} - \mathbf{X}^T\mathbf{w}\right)\right) + \lambda\mathbf{w}^T\mathbf{w}\right] \quad (1)$$

where $\mathbf{F} = diag(f'(\bar{d}_1), f'(\bar{d}_2), \ldots, f'(\bar{d}_n))$ is a diagonal matrix formed by the derivative of the f function for the components of $\bar{\mathbf{d}}$, and λ is a positive scalar that controls the influence of the penalty term in the solution. If the hyperparameter λ is set to zero then the basic MSE is obtained.

The cost function defined in Eq. (1) has the advantage that, despite using nonlinear activation functions (f), it is convex and the global optimum can be obtained directly by means of a closed solution for a given $\mathbf{X}$ and $\mathbf{d}$. Specifically, the minimum of this cost function can be obtained by deriving it and equating the result to zero. The solution is the $\mathbf{w}$ that satisfies the following system of linear equations [41]:

$$(\mathbf{XFFX}^T + \lambda\mathbf{I})\mathbf{w} = \mathbf{XFF}\bar{\mathbf{d}} \quad (2)$$

The size of the system of linear equations in Eq. (2) depends on m, as the most costly operations to solve the equations and obtain $\mathbf{w}$ are calculations of matrix $(\mathbf{XFFX}^T + \lambda\mathbf{I})$ and its inverse, with computational complexity of $O(m^2n)$ and $O(m^3)$, respectively. When m is small, these complexities imply high efficiency, but when the number of inputs is large, they become computationally demanding, even though the method provides a non-iterative way to determine $\mathbf{w}$. In order to obtain $\mathbf{w}$ in the most efficient way possible, irrespective of whether the data contains a greater number of samples than variables or vice versa, Eq. (2) is transformed [41] using singular value decomposition (SVD). This results in a factorization of matrix $\mathbf{XF} = \mathbf{USV}^T$, where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and $\mathbf{S} \in \mathbb{R}^{m\times n}$ is a diagonal matrix with r non-zero elements, known as the singular values of $\mathbf{S}$, where $r = rank(\mathbf{XF}) \leq min(m, n)$. Using this approach Eq. (2) can be rewritten as follows:

$$(\mathbf{USV}^T\mathbf{FX}^T + \lambda\mathbf{I})\mathbf{w} = \mathbf{XFF}\bar{\mathbf{d}} \quad (3)$$

whose optimal solution, i.e., that which provides the minimum error for a given training set, can also be obtained by a closed-form as follows [41]:

$$\mathbf{w} = \mathbf{U}(\mathbf{SS}^T + \lambda\mathbf{I})^{-1}\mathbf{U}^T\mathbf{XFF}\bar{\mathbf{d}} \quad (4)$$

As the number of non-zero elements in the diagonal matrix $\mathbf{S}$ is r, the effective dimensions of $\mathbf{U}$ and $\mathbf{S}$ are $m \times r$ and $r \times r$, respectively. This allows us to calculate SVD($\mathbf{XF}$) using an economy-sized decomposition that contains all the relevant information of the regular decomposition, but as it is more compact, it can be calculated more efficiently in time $O(mnr)$ [42]. Moreover, as long as λ is non-zero, $(\mathbf{SS}^T + \lambda\mathbf{I})$ is a diagonal matrix with non-zero elements in the diagonal, and, therefore, the calculation of the inverse lowers complexity to time $O(r)$; this is because only the reciprocals of the main diagonal elements have to be calculated. Under these circumstances, as $r \leq min(m, n)$, the system in Eq. (3) becomes efficient, irrespective of whether $m \gg n$ or $n \gg m$.

3.2. Federated and homomorphically encrypted learning method (FedHEONN)

The solution presented in Eq. (4) is only applicable in a centralized learning scenario, where the entire data set $\mathbf{X}$ is accessible in a single location. However, in the case of a federated environment, the data is partitioned into P submatrices $\mathbf{X} = [\mathbf{X}_1|\mathbf{X}_2|\ldots|\mathbf{X}_P]$, such that each node (client) contains only one of the submatrices. This is a type of FL known as horizontal learning, in which it is assumed that all clients share the same variables for the problem but differ in the samples available. FedHEONN reformulates the research described in Section 3.1, taking into account the following considerations regarding the solution presented in Eq. (4):

- The term $\mathbf{U}(\mathbf{S}\mathbf{S}^T + \lambda\mathbf{I})^{-1}\mathbf{U}^T$ involves the matrices $\mathbf{U}$ and $\mathbf{S}$, which, in the original algorithm, are calculated from the centralized SVD of $\mathbf{XF}$. However, Iwen and Ong [43] demonstrated that the SVD can be computed in an incremental and distributed way. Given a data matrix $\mathbf{A}$, decomposed into P partitions $\mathbf{A} = [\mathbf{A}_1|\mathbf{A}_2|\dots|\mathbf{A}_P]$, this has the same singular values $\mathbf{S}$ and left singular vectors $\mathbf{U}$ as the matrix $\mathbf{B} = [\mathbf{U}_1\mathbf{S}_1|\mathbf{U}_2\mathbf{S}_2|\dots|\mathbf{U}_P\mathbf{S}_P]$, where $\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p = \mathrm{SVD}(\mathbf{A}_p),\ \forall p = 1,\dots,P$, i.e:
$$\begin{aligned}\mathrm{SVD}(\mathbf{A}) &= \mathrm{SVD}([\mathbf{A}_1|\mathbf{A}_2|\dots|\mathbf{A}_P])\\ &= \mathrm{SVD}([\mathbf{U}_1\mathbf{S}_1|\mathbf{U}_2\mathbf{S}_2|\dots|\mathbf{U}_P\mathbf{S}_P]) \qquad (5)\end{aligned}$$
Therefore, applying the incremental method, we can compute the economy-sized SVD of $\mathbf{XF}$ using the partial SVDs calculated across all clients in the federated scenario.
This partial SVD merging scheme has been shown to be numerically robust to rounding off errors and corruption of the original data. It is also accurate even when the rank of matrix $\mathbf{A}$ is underestimated or deliberately reduced.
- The term $\mathbf{XF}\mathbf{F}\bar{\mathbf{d}}$, which, for simplicity sake, we call $\mathbf{m} \in \mathbb{R}^{m\times 1}$, can be computed by means of a block partitioned matrix product that only involves algebra in the submatrices of the factors, using the following equation:
$$\begin{aligned}\mathbf{m} = \mathbf{XF}\mathbf{F}\bar{\mathbf{d}} &= [\mathbf{X}_1|\mathbf{X}_2|\dots|\mathbf{X}_P]\begin{bmatrix}\mathbf{F}_1\\ \mathbf{F}_2\\ \vdots\\ \mathbf{F}_P\end{bmatrix}\begin{bmatrix}\mathbf{F}_1\\ \mathbf{F}_2\\ \vdots\\ \mathbf{F}_P\end{bmatrix}\begin{bmatrix}\bar{\mathbf{d}}_1\\ \bar{\mathbf{d}}_2\\ \vdots\\ \bar{\mathbf{d}}_p\end{bmatrix}\\ &= \mathbf{X}_1\mathbf{F}_1\mathbf{F}_1\bar{\mathbf{d}}_1 + \dots + \mathbf{X}_p\mathbf{F}_p\mathbf{F}_p\bar{\mathbf{d}}_p \qquad (6)\end{aligned}$$
Therefore, $\mathbf{m}$ can be incrementally computed through the local information provided by P clients. Let us suppose that, at client p, we have a local data set of n_p examples $\mathbf{X}_p \in \mathbb{R}^{m\times n_p}$ and that $\mathbf{m}_p = \mathbf{X}_p\mathbf{F}_p\mathbf{F}_p\bar{\mathbf{d}}_p$ is computed.
When new data, $\mathbf{X}_k$, from another client k become available, the new vector $\mathbf{m}_{p|k}$ – which combines the information provided jointly by $\mathbf{X}_p$ and $\mathbf{X}_k$ – can be easily obtained by simply adding to $\mathbf{m}_p$, the term that depends only on the new data block, i.e:
$$\mathbf{m}_{p|k} = \mathbf{m}_p + \mathbf{X}_k\mathbf{F}_k\mathbf{F}_k\bar{\mathbf{d}}_k \qquad (7)$$
- The proposed method is private by design, as no raw data is sent across the network to aggregate the information from multiple clients, only locally computed matrices $\mathbf{U}_p\mathbf{S}_p$ and $\mathbf{m}_p$. However, we use an HE mechanism to add an additional protective barrier against attacks aimed at causing privacy leaks by inferring the data used during the training process, such as happens with model inversion attacks. The advantage of this type of encryption, as previously mentioned, is that it allows us to operate with the encrypted data, while obtaining the same result as if the operation had been performed on the plain data. We introduce encryption in the $\mathbf{m}_p$ vector sent by each client to the coordinator; this information may be more vulnerable to attacks than the $\mathbf{U}_p\mathbf{S}_p$ matrix, since the latter comes from the calculation of the SVD of $\mathbf{X}_p\mathbf{F}_p$ and part of the information necessary to recover the data (the matrix $\mathbf{V}_p$ of the factorization) is never sent.
Therefore, the $\mathbf{m}_p$ vector is encrypted before being sent to the coordinator in charge of aggregating all client information. Subsequently, the coordinator adds the information provided by client p to the global vector $\mathbf{m}$ by means of the following homomorphic addition operation:
$$[\![\mathbf{m}]\!] = [\![\mathbf{m}]\!] + [\![\mathbf{m}_p]\!] \qquad (8)$$
where $[\![\cdot]\!]$ is the HE operator. We used a CKKS HE scheme [44], since, unlike other HE schemes, it supports approximate arithmetic over real numbers; although it only supports homomorphic addition and homomorphic multiplication, these operations are enough to implement the proposed method. This mechanism provides a higher level of security against inversion model attacks and non-trust worthy coordination servers, as the operations are performed using encrypted matrices.

Considering the above, we propose a new FL model with HE in a client–server setup, where P participants (clients) with the same data structure collaboratively train an ML model with the help of an aggregation server (coordinator). In order to perform FL at client p, using the local data partition $\mathbf{X}_p$, we only need to compute the matrices $\mathbf{U}_p$, $\mathbf{S}_p$ resulting from the SVD of $\mathbf{X}_p\mathbf{F}_p$, vector $\mathbf{m}_p = \mathbf{X}_p\mathbf{F}_p\mathbf{F}_p\bar{\mathbf{d}}_p$, and its encrypted version $[\![\mathbf{m}_p]\!]$ using the CKKS HE scheme. Each client subsequently sends their partial results to the coordinator, which incrementally combines all the information using Eqs. (5) and (8) to obtain the global matrices $\mathbf{m}$, $\mathbf{U}$ and $\mathbf{S}$. Weights are then calculated using Eq. (4). Since no raw data is transmitted between nodes the algorithm guarantees privacy.

In the FL scheme, the 1 and 2 algorithms contain the pseudocode for the clients and the coordinator, respectively. The pseudocode for the coordinator is designed to receive the list of matrices and vectors calculated by all the available clients, and to aggregate them all sequentially. Although the method can deal with all clients in a single stage, the coordinator can also add clients at different stages if any are temporarily unavailable. To do this, the coordinator starts aggregating the information provided by the available clients and, incrementally adds new clients. This also allows client addition dynamically, since it is not necessary to retrain previously aggregated clients to add the information provided by the new client. In the interest of contributing to reproducible research, the Python code is available online.[1]

Algorithm 1 Pseudocode for the FedHEONN client

```
Inputs for a client p:
    $\mathbf{X}_p \in \mathbb{R}^{m\times n_p}$      ▷ Local data block with $m$ inputs and $n_p$ samples
    $\mathbf{d}_p \in \mathbb{R}^{n_p\times 1}$      ▷ The corresponding local vector of desired outputs
    $f$                                             ▷ Nonlinear activation function (invertible)

Outputs:
    $[\![\mathbf{m}_p]\!]$        ▷ Encrypted local $\mathbf{m}$ vector computed by client $p$
    $\mathbf{US}_p$               ▷ Local $\mathbf{U} * \mathbf{S}$ matrix computed by client $p$

 1: function FEDHEONN_CLIENT($\mathbf{X}_p$,$\mathbf{d}_p$,$f$)
 2:     $\mathbf{X}_p = [ones(1, n_p); \mathbf{X}_p]$;                 ▷ Bias is added
 3:     $\bar{\mathbf{d}}_p = f^{-1}(\mathbf{d}_p)$;                  ▷ Inverse of the neural function
 4:     $\mathbf{f}_p = f'(\bar{\mathbf{d}}_p)$;                      ▷ Derivative of the neural function
 5:     $\mathbf{F}_p = diag(\mathbf{f}_p)$;                          ▷ Diagonal matrix
 6:     $[\mathbf{U}_p, \mathbf{S}_p, \sim] = SVD(\mathbf{X}_p * \mathbf{F}_p)$;   ▷ Economy size SVD
 7:     $\mathbf{US}_p = \mathbf{U}_p * diag(\mathbf{S}_p)$           ▷ Local product $\mathbf{US}_p$ is computed
 8:     $\mathbf{m}_p = \mathbf{X}_p * (\mathbf{f}_p .* \mathbf{f}_p .* \bar{\mathbf{d}}_p)$;   ▷ Local vector $\mathbf{m}_p$ is computed
 9:     $[\![\mathbf{m}_p]\!]$ = ckks_encryption($\mathbf{m}_p$)     ▷ CKKS encryption of the $\mathbf{m}_p$ vector
10:     return $[\![\mathbf{m}_p]\!]$, $\mathbf{US}_p$
11: end function
```

[1] https://github.com/ofontenla/FedHEONN.

Algorithm 2 Pseudocode for the FedHEONN coordinator

```
Inputs:
    M_list    ▷ List containing the encrypted M matrices of the available clients
    US_list   ▷ List containing the US matrices of the available clients
    $\lambda$   ▷ Regularization hyperparameter

Outputs:
    $[\![\mathbf{w}]\!] \in \mathbb{R}^{m\times 1}$   ▷ Encrpyted optimal weights

1:  function FEDHEONN_COORDINATOR(M_list, US_list, $\lambda$)
2:      if previous $[\![\mathbf{m}]\!]$, US matrices are available:
3:          $[\![\mathbf{m}]\!]$ = Stored $[\![\mathbf{m}]\!]$   ▷ The vector is initialized with the existing one
4:          US = Stored US   ▷ The matrix is initialized with the existing one
5:      else:   ▷ First time that clients are aggregated
6:          $[\![\mathbf{m}]\!]$ = 0   ▷ Zero vector
7:          US = [ ]   ▷ Empty matrix
8:      for $[\![\mathbf{m}_p]\!]$, $\mathbf{US}_p$ in (M_list, US_list):   ▷ Loop through clients
9:          $[\![\mathbf{m}]\!] = [\![\mathbf{m}]\!] + [\![\mathbf{m}_p]\!]$   ▷ Aggregation of m vector
10:         $[\mathbf{U}, \mathbf{S}, \sim] = SVD([\mathbf{US} \mid \mathbf{US}_p])$;   ▷ Incremental SVD
11:         $\mathbf{US} = \mathbf{U} * diag(\mathbf{S})$   ▷ Aggregation of US Matrix
12:     $[\![\mathbf{w}]\!] = \mathbf{U} * inv(\mathbf{S}*\mathbf{S} + \lambda\mathbf{I}) * (\mathbf{U}^T * [\![\mathbf{m}]\!])$   ▷ Optimal weights
13:     Save $[\![\mathbf{m}]\!]$, US   ▷ To aggregate new information in the future
14:     return $[\![\mathbf{w}]\!]$
15: end function
```

Note that the CKKS implementation of the HE has a limit on the number of multiplications performed on encrypted data, since each operation adds noise, and once a certain amount of noise is added it is no longer possible to retrieve the original data. Specifically, recent studies [45] have verified that, for large sizes of encrypted vectors, no more than two sequential multiplications should be done, so as not to degrade the result obtained with encryption. However, this does not imply a limitation for our proposed method, since the chained operations are performed on the addition operations (line 9 of algorithm 2), and multiplications are only performed on two consecutive operations (line 12 of algorithm 2), regardless of the number of clients. As detailed below in the results section, this has also been verified experimentally in that the accuracy of the model is not degraded.

In addition, an important feature of the method is that, unlike most of the state-of-the-art federated methods, it requires only a single round of coordinator aggregation involving all available clients. This is because, as a non-iterative learning method, the optimal weights for the clients can be obtained in a single training step. In any case, if any of the clients receive new data in the future, this knowledge can be very easily added to the previously trained model.

As a summary, Fig. 1 depicts the main phases of the proposed method.

3.3. Communication efficiency

In this subsection, we analyze the total amount of information sent through the communication network during training, both by the proposed method and by a traditional federated method, such as FedAVG, which uses several training rounds with a fraction of the clients participating per round. In both cases, the same base model (one-layer network with one output) is employed. The analysis is based on the following definitions and considerations:

- c: number of clients in the federated environment.
- m: number of features in the data set.
- n: total number of data instances. In the federated environment, $n = \sum_{i=1}^{c} n_i$, where n_i is the number of instances in the local data set of client i.

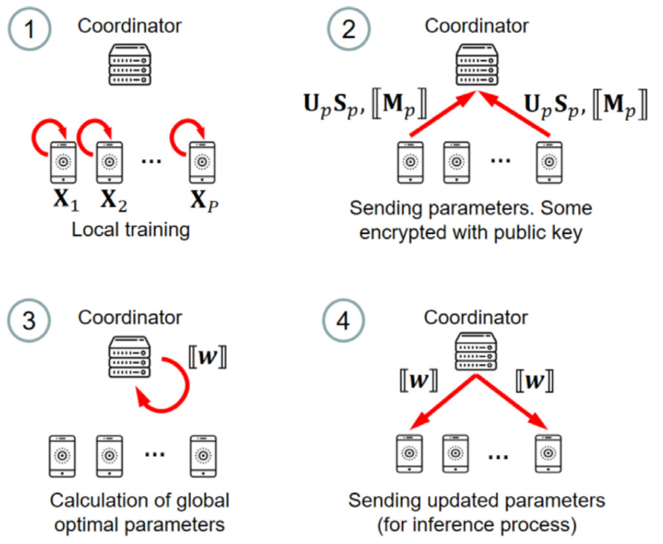
1
Coordinator
$\mathbf{X}_1$
$\mathbf{X}_2$
...
$\mathbf{X}_P$
Local training
2
Coordinator
$\mathbf{U}_p\mathbf{S}_p, [\![\mathbf{M}_p]\!]$
$\mathbf{U}_p\mathbf{S}_p, [\![\mathbf{M}_p]\!]$
Sending parameters. Some encrypted with public key
3
Coordinator
$[\![\boldsymbol{w}]\!]$
Calculation of global optimal parameters
4
Coordinator
$[\![\boldsymbol{w}]\!]$
$[\![\boldsymbol{w}]\!]$
Sending updated parameters (for inference process)

Fig. 1. Main steps of the proposed federated method.

- ρ: fraction of clients employed in each round ($0 < \rho \leq 1$). In the proposed method, ρ is always equal to 1, since there is only one round of communication in which all clients participate.
- The traditional federated method uses r communication rounds, and for each round it sends the weights of the network (a vector of size m).
- In the proposed method the information sent by the communication network is the matrix $\mathbf{US}_i \in \mathbb{R}^{m\times k_i}$, with $k_i = min(m, n_i)$, and the vector $[\![\mathbf{m}_i]\!] \in \mathbb{R}^m$.

Using the above statements, the information sent can be computed by FedHEONN ($I_{FedHEONN}$) as follows:

$$I_{FedHEONN} = \sum_{i=1}^{c}(\overbrace{mk_i}^{\text{size of } \mathbf{US}_i} + \overbrace{m}^{\text{size of } \mathbf{m}_i}) \tag{9}$$

and by the traditional federated method (I_{Fed}) as follows:

$$I_{Fed} = \rho cmr \tag{10}$$

The following operations can be carried out to find out when the quantity of information sent by the proposed method is less than that of the traditional method:

$$I_{FedHEONN} < I_{Fed} \tag{11}$$

$$\sum_{i=1}^{c}(mk_i + m) < \rho cmr \tag{12}$$

$$\sum_{i=1}^{c}(k_i + 1) < \rho cr \tag{13}$$

$$\sum_{i=1}^{c} k_i + c < \rho cr \tag{14}$$

$$\frac{\sum_{i=1}^{c} k_i + c}{\rho c} < r \tag{15}$$

Using Eq. (15) it can be concluded that the communications in the proposed method are more efficient when the number of traditional FL method rounds r satisfies that:

$$r > \frac{\frac{1}{c}\sum_{i=1}^{c} k_i + 1}{\rho} \tag{16}$$

In addition, using this equation we can analyze two main cases:

(a) if $m \leq n_i$, $\forall i = 1 \ldots c$, then using Eq. (16) we get the following result:

$$\frac{\frac{1}{c}\sum_{i=1}^{c} k_i + 1}{\rho} < r \tag{17}$$

$$\frac{\frac{1}{c}\sum_{i=1}^{c} m + 1}{\rho} < r \tag{18}$$

$$\frac{\frac{cm}{c} + 1}{\rho} < r \tag{19}$$

$$\frac{m + 1}{\rho} < r \tag{20}$$

(b) if $m > n_i$, $\forall i = 1 \ldots c$, then using Eq. (16) we get the following result:

$$\frac{\frac{1}{c}\sum_{i=1}^{c} k_i + 1}{\rho} < r \tag{21}$$

$$\frac{\frac{1}{c}\sum_{i=1}^{c} n_i + 1}{\rho} < r \tag{22}$$

$$\frac{\frac{n}{c} + 1}{\rho} < r \tag{23}$$

4. Results

In this section, we evaluate the performance of the proposed FeHEONN method, in terms of accuracy and consumption, by testing it on different federated configurations and analyzing the influence of the number of clients on performance. To verify that the federated approach does not imply performance degradation, we compare the results with the centralized version of the algorithm, with just a single client containing all the data. In addition, to analyze the impact of encryption, we include results with and without encryption. Finally, given the importance of data distribution in federated environments, our experiments use both IID and non-IID data.

After analyzing FedHEONN performance, we performed a brief experiment to compare it with FedAVG (the most widely used approach in the FL environment) in terms of accuracy and efficiency of CPU usage and communications involved in training. We also studied test accuracy for our method compared to other state-of-the-art ML methods.

4.1. Experimental setup

To evaluate performance in a federated environment as close as possible to reality, we performed different experiments based on varying the number of clients from 200 to 20,000 in increments of 200. We selected six large data sets, so that, in the most extreme case (20,000 clients), each client could count on a few data points. Table 1 shows the main characteristics of the classification data sets used, all representing real-world problems and publicly available at the UCI data set repository [46]. Only the DryBean×10 and Higgs×4 data sets were semi-artificially created by replicating ten and four times the data from the Dry Bean and Higgs data sets, respectively. While this experiment was not relevant in terms of accuracy, it is interesting to analyze training time for the method in an even more extreme scenario.

In all experimental tests, the data were divided into two sets: a training set made up of 70% of the data and a test set containing the remaining 30%. In addition, to create the federated scenario, the training data were distributed among all the clients so that each one had approximately the same amount of data. Table 2 shows the size of the centralized training set and the approximate size of the local training datasets available at each client in the most extreme federated scenario (20,000 clients). As can be seen, in this extreme scenario, the amount of local data in each client was very small.

Table 1
Characteristics of the data sets.

Dataset	Samples	Attributes	Classes
MiniBoone	130,064	50	2
DryBean×10	136,110	16	7
Skin	245,057	3	2
SUSY	5,000,000	18	2
HEPMASS	10,500,000	28	2
Higgs	11,000,000	28	2
Higgs×4	44,000,000	28	2

Table 2
Size of the training set at each client for the centralized scenario and the most extreme federated scenario (20,000 clients).

Dataset	# samples	
	Global training set	Local training set
MiniBoone	91,044	4
DryBean×10	95,277	4
Skin	171,539	8
SUSY	3,500,000	175
HEPMASS	7,350,000	368
Higgs	7,700,000	385
Higgs×4	30,800,000	1,540

We carried out all experiments for IID and non-IID scenarios and compared the results with the same method but without the encryption layer. To create the non-IID scenario, the global data set was ordered according to the class label, and local data sets were created for each client following the pre-established order. This was a pathological non-IID partition of the data, as the vast majority of the clients would only have instances of one of the classes.

In all cases, the regularization hyperparameter (λ) was 1×10^{-3}. No tests were carried out with different values of this hyperparameter since the aim was to observe the behavior of the model by only varying the number of clients. Finally, logistic functions were used as activation functions for neurons.

Since 20,000 different machines were not available to run a client on each, the federated environment was simulated on the same computer. All the clients of the federated model and also the centralized model were executed on a computer with an Intel Core i7-10700 2.90 GHz processor with 32 GB of RAM.

The following metrics were used to analyze model performance:

- Test accuracy: Correctly predicted data points as a percentage of all the data points in the test set.
- Training time: In a real federated environment, since clients run in parallel on different devices, the total training time required to obtain the federated model is the time taken by the slowest client (the last to send information to the coordinator) plus the time taken by the coordinator to aggregate the information from all clients.
- Sum of CPU time: The individual training times of all the clients plus the time of the coordinator. This metric is an important reference reflecting the overall energy consumption of the federated system.
- Watts per hour (Wh): Consumption calculated in terms of Wh is based on the nominal watts consumed by the CPUs of the devices and the CPU time used by each device for training. In this case, as all clients use the same type of device, the calculation is the result of multiplying the watts by the sum of CPU time (in seconds) divided by 3,600.

4.2. Performance results

In this section the results of the proposed method are analyzed and compared with the results for the same model without applying encryption. The goal of this comparison is to analyze whether or not operations with HE information significantly degrade the model's accuracy. The simulations were repeated three times. Note that the reported results refer to the mean value for each of the metrics.

Fig. 2 shows results for the test accuracy of the method with and without encryption, varying the number of clients in IID and non-IID scenarios. The first value of the curves i.e., *x*-axis value 1, corresponds to centralized training of the model. It can be observed that, regardless of the number of clients, accuracy is maintained with very little variability. Even in extreme scenarios in which very little local data are available for 20,000 clients (4 or 8 data points for MiniBoone, DryBean$\times$10 and Skin), model behavior is stable. It can be concluded that the proposed method is very robust in terms of accuracy in all scenarios, as it is not influenced by the number of clients, the use of encryption, or non-identical distributed data.

Fig. 3 shows the method training time for the various scenarios considered. Analyzing the plots, the following can be observed:

- As expected, the model with encryption supposed an additional computational cost compared to the model without encryption. Even so, for large data sets (SUSY, HEPMASS, Higgs, and Higgsx4) and up to a relatively high number of clients, the proposed method obtained significantly lower training times than the centralized version without encryption (x-axis value 1). As an example, the average training time of the centralized model for Higgsx4 was 464.84 s compared to 11.18 s for the proposed method with 10,000 clients.
- Due to the computational cost of adding all the information in the coordinator, FedHEONN training time grew linearly up to a certain number of clients, when the computational cost starts to increase significantly due to the accumulation of encrypted operations in the coordinator. As can be seen, this change in trend did not occur in the non-encrypted case.

Fig. 4 shows the sum of CPU time consumed by all clients and the coordinator (left vertical axis), and also shows the corresponding Wh (right vertical axis). As can be observed, in terms of Wh, encryption supposed, in general, an extra cost. This is to be expected since it is necessary to carry out data encryption and operate with encrypted data, which always has a higher computational cost than an operation with plain data. Despite this, for large data sets, such as Higgsx4 or even much larger data sets, certain FedHEONN federated environment configurations were more efficient in terms of power consumption, and were therefore more sustainable. More specifically, for the Higgsx4 dataset, power consumption for any of the scenarios between 2 and approximately 8,000 clients was lower than the centralized model (without encryption), a fact that would be generalizable and expandable to even much larger data sets.

4.3. Comparison with FedAVG

To demonstrate whether our method, FedHEONN, is comparable with FedAVG, the most widely used method in the FL environment, experiments were performed using the SUSY data set. We employed the same base model (one-layer neural network) for FedAVG and for FedHEONN. In this case, the number of clients was varied from 5 up to 50, in increments of 5. For all the experiments, FedAVG used 100 communication rounds with different fractions of clients (0.1, 0.3 and 0.5).

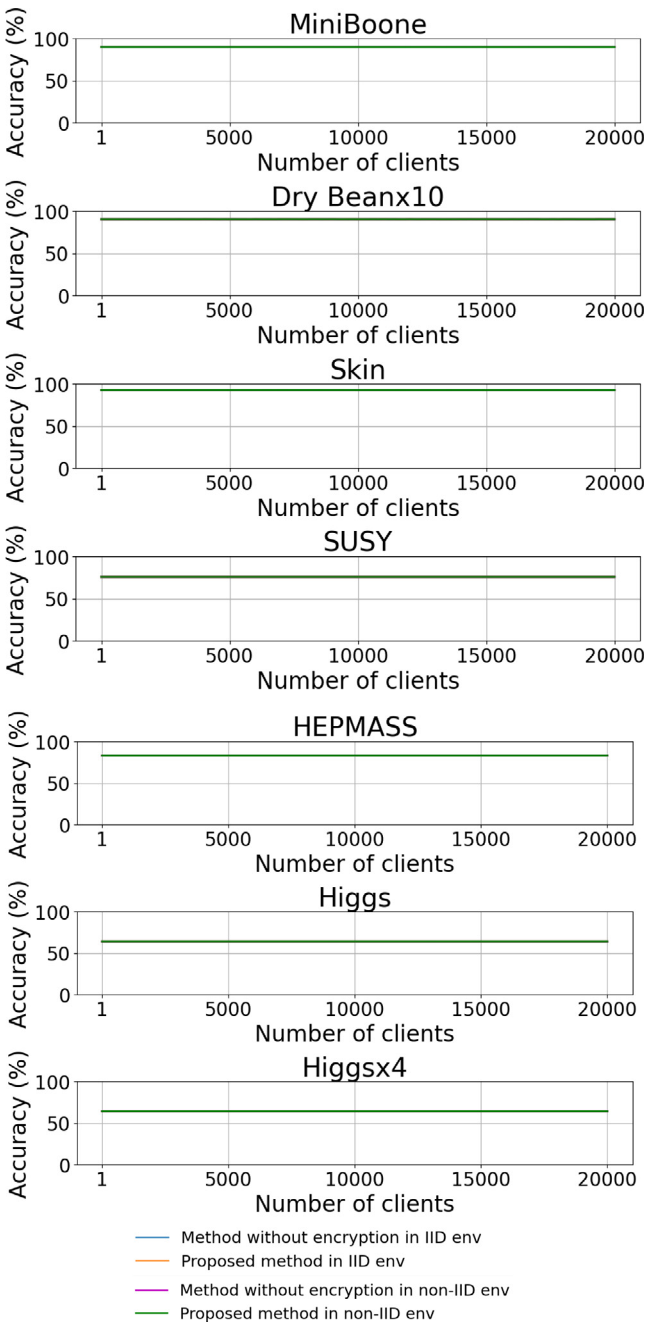

Fig. 2. Mean test accuracy as a function of the number of clients, for the IID and non-IID scenarios, with and without encryption. Since the four curves in each subplot overlap, only one can be seen.

In terms of test accuracy stability, FedAVG showed similar behavior and its accuracy did not vary regardless of the number of clients employed or the fraction of clients used for each round.

With regards to training time, Fig. 5 shows that, despite the addition of an encryption stage, time for the FedHEONN model is much lower time than for the FedAVG, irrespective of the configuration.

Fig. 6 depicts the *sum of CPU time* and *Wh* consumed in the considered scenarios, clearly showing that FedHEONN's power consumption is the slowest.

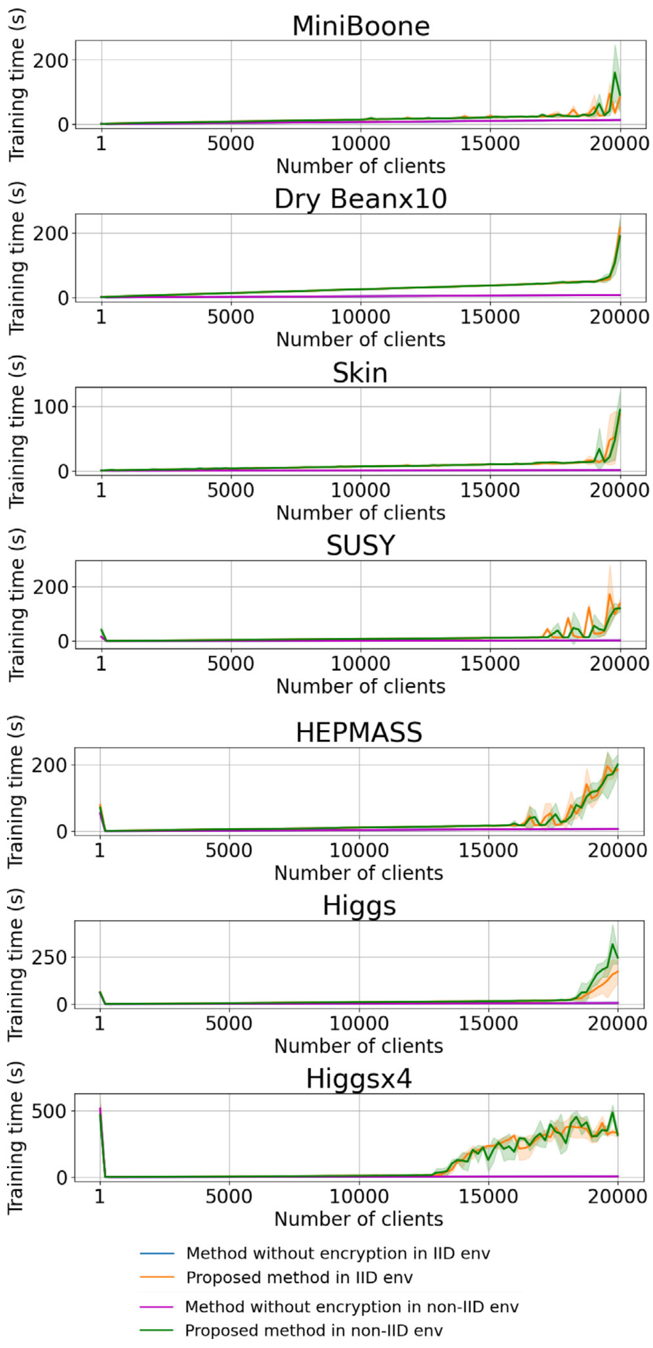

Fig. 3. Mean training time as a function of the number of clients, for the IID and non-IID scenarios, with and without encryption. The shaded areas reflect the standard deviation of each curve.

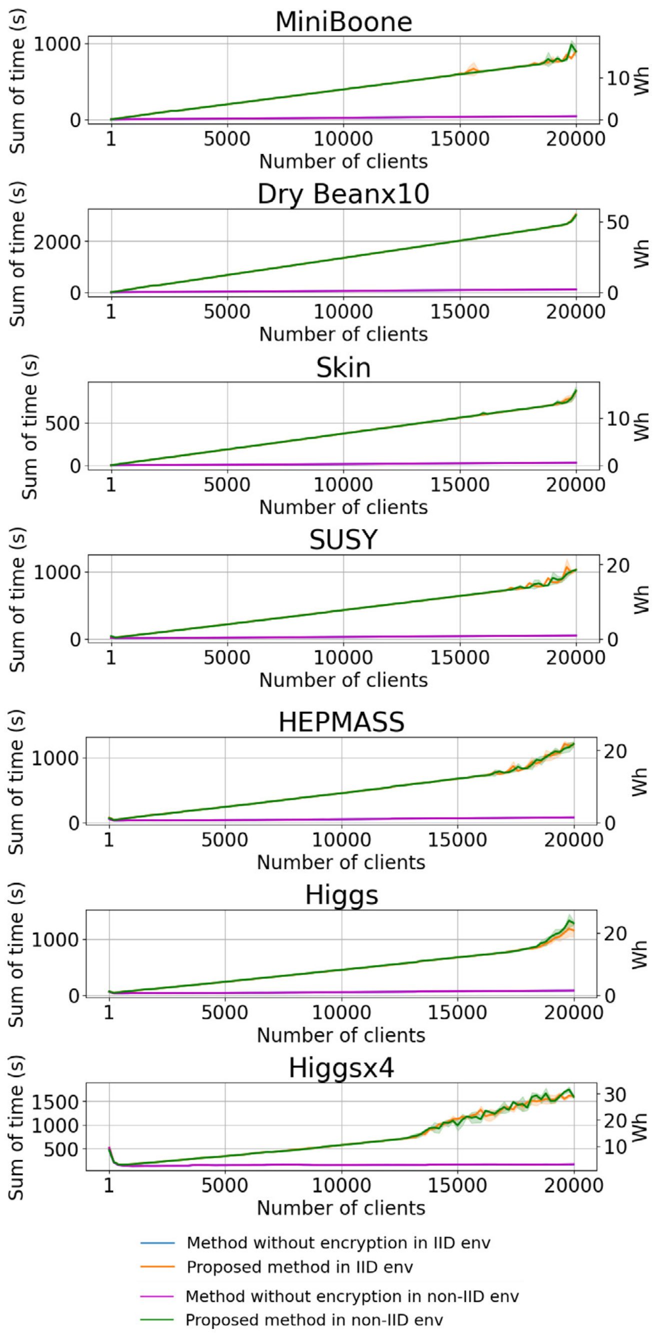

Fig. 4. Mean sum of CPU time and Wh consumed in training as a function of the number of clients for the IID and non-IID scenarios, with and without encryption. The shaded areas reflect the standard deviation of each curve.

4.4. Accuracy comparison with other state-of-the-art results

We include a comparative study of the test accuracy obtained by FedHEONN with respect to the accuracy obtained by other state-of-the-art LM models and a single layer neural network that is like our model, but trained using all the data centralized in a single site and with an iterative training algorithm (quasi-Newton method). Tables 3 and 4 contain the quantitative results obtained by FedHEONN together with other results found in the scientific literature. The first column of the table shows the name of the method and the corresponding bibliographic reference. Hyphenated cells indicate that the result is not available.

For ease of comparison, mean and median accuracy (calculated excluding FedHEONN) are reported at the end of the tables. As can be seen, FedHEONN, despite being based on a network model without hidden layers, obtained very competitive results, even against more complex ML models that also use all the data in a centralized way, and achieved better results in many cases. Furthermore, a comparison of the overall results shows that FedHEONN was better in 5 out of 6 cases when considering the mean, and in 3 out of 6 cases when considering the median, all with the added advantage that it can be used in a federated environment (IID or non-IID) with very low power consumption for the processing of large data sets.

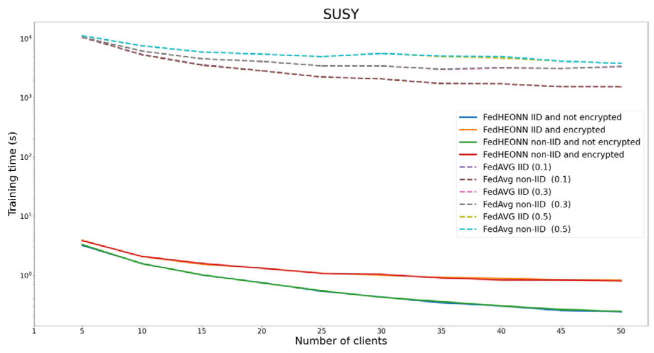

Fig. 5. Training time for the SUSY data set as a function of the number of clients, for IID and non-IID scenarios. Considered for FedHEONN were both encrypted and unencrypted versions and for FedAVG different fractions of clients per round. The y-axis is represented in log scale.

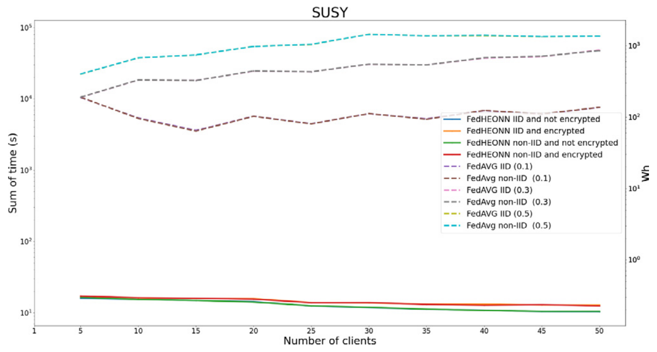

Fig. 6. Sum of CPU time and Wh consumed in training the SUSY data set as a function of the number of clients for the IID and non-IID scenarios. Considered for FedHEONN were both encrypted and unencrypted versions and for FedAVG different fractions of clients per round. The y-axis is represented in log scale.

4.5. Communication efficiency

Finally, using the study presented in Sub Section 3.3 regarding the amount of information sent through the communications network, we analyzed the results obtained for the data sets for the scenario with the largest number of clients ($c = 20{,}000$). Fig. 7 shows the number of rounds, as a function of ρ, from which the FedHEONN method was more efficient in terms of the amount of information sent than a traditional federated method employing the same base model. Note that the Higgs and HEPMASS curves overlap, so only the latter is visible. As can be seen, FedHEONN is more efficient than the traditional method in a low number of communication rounds. This difference became more acute as the fraction of participants per round (ρ) in the traditional method increased. As an example, for the Skin data set and considering a fraction of clients (ρ) of 0.3, FedHEONN was more efficient than the other federated model when the latter used more than 13 rounds of communication.

5. Conclusions

In this work, we have described FedHEONN, a novel federated ML method, based on one-layer neural networks, that incorporates HE. It has the following advantages compared to current FL methods:

- FedHEONN obtains equivalent solutions irrespective of whether the environment is IID or non-IID. This is an important advantage since most FL models do not yield the same solutions and performance in both situations.

Table 3
Accuracy (%) results for FedHEONN compared with results published by other authors for the small data sets.

Model	Dry Bean	MiniBoone	Skin
Proposed method	**90.43**	**90.02**	**92.56**
Single layer neural network	92.67	90.46	91.90
Logistic regression [47]	91.00	–	–
K-nearest neighbor [47]	89.00	–	–
K-nearest neighbor [48]	92.52	–	–
Condensed Nearest Neighbor [49]	–	82.96	99.95
Editing Nearest Neighbor [49]	–	87.59	99.97
Reduced Nearest Neighbor [49]	–	–	99.95
Multilayer perceptron [47]	91.00	–	–
Multilayer perceptron [48]	91.73	–	–
Artificial Neural Network [50]	92.58	–	–
Support Vector Classifier [47]	92.00	–	–
Support Vector Classifier [48]	93.13	–	–
Support Vector Classifier [50]	92.18	–	–
Naive Bayes [47]	89.00	–	–
Multinomial Naive Bayes [50]	64.30	–	–
Decision Tree [47]	90.00	–	–
Decision Tree [48]	87.92	–	–
Decision Tree [50]	91.59	–	–
Very Fast Decision Tree [51]	–	–	93.33
Hybrid Decision Tree [51]	–	–	98.42
C4.5 [51]	–	–	99.48
Random Forest [47]	92.00	–	–
Random Forest [50]	93.61	–	–
ELM [49]	–	91.70	99.27
ELM-EN [49]	–	91.08	99.25
ELM-KL [49]	–	88.62	98.08
ELM-KL-ALL [49]	–	92.07	98.89
Difference of Convex Algorithm DCA [52]	–	84.19	–
Stochastic DCA [52]	–	83.90	–
Sequential-based DL Model [53]	94.88	–	–
Extreme Gradient Boosting [47]	93.00	–	–
Voting Classifier [50]	92.80	–	–
Mean	**90.35**	**88.06**	**98.04**
Median	**92.00**	**88.62**	**99.25**

Table 4
Accuracy (%) results for FedHEONN compared with results published by other authors for the large data sets.

Model	Higgs	SUSY	HEPMASS
Proposed method	**64.05**	**75.76**	**83.50**
Single layer neural network	64.17	78.80	83.64
Logistic Regression [54]	64.21	–	–
Logistic Regression [55]	–	78.84	–
Spark K-means [56]	48.34	50.04	50.66
Spark Generalized Linear Model [56]	63.51	75.01	83.40
Decision Tree [54]	63.57	–	–
Decision Tree [55]	–	75.46	–
Random Forest [54]	67.64	–	–
Random Forest [55]	–	77.40	–
Random Forest [57]	67.67	77.67	82.21
Spark Random Forest [56]	59.65	76.81	82.43
Rotation Forest [57]	68.80	78.59	84.44
Gradient Boosted Tree [54]	70.62	–	–
Gradient Boosted Tree [55]	–	79.30	–
Spark Gradient Boosted Tree [56]	59.49	75.11	81.83
PANFIS [58]	63.94	75.42	83.32
PANFIS MapReduce [58]	63.48	76.80	83.35
Scalable PANFIS Merging [56]	63.66	76.70	83.47
Scalable PANFIS Voting [56]	63.70	76.22	84.18
Scalable PANFIS AL Merging [56]	63.72	76.79	83.45
Scalable PANFIS AL Voting [56]	63.92	76.20	84.15
PDMS Genetic Algorithm [59]	63.00	–	83.67
Max Mean Discrepancy [60]	57.90	–	–
eTS [58]	64.69	77.05	82.32
Simpl_eTS [58]	60.17	70.93	81.22
MapReduce MRAC [61]	62.96	74.57	–
PCA Random Discretization Ensemble [57]	58.33	72.64	81.33
Mean	**62.92**	**75.06**	**81.12**
Median	**63.66**	**76.70**	**83.35**

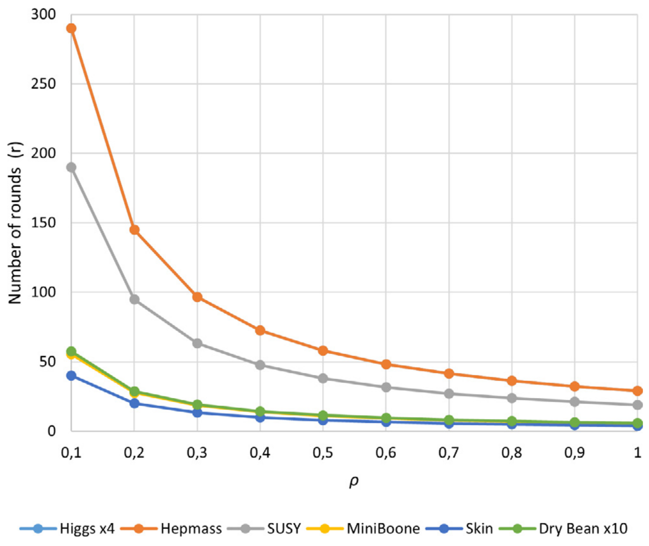

Fig. 7. Number of rounds for which FedHEONN communications are more efficient than the traditional federated method, as a function of ρ.

- For FedHEONN, only a single round of training involving all the devices is necessary to obtain the optimal weights of the network, allowing a faster learning process than produced by the vast majority of current methods in the literature, which use multiple rounds of training in which part of the clients collaborate.
- FedHEONN is suitable for working in both cross-silo and cross-device settings.
- FedHEONN, in addition to being a privacy-preserving algorithm by design, because it does not send raw private data directly over the communication network, it is also robust against model inversion attacks since the local clients send the computed vectors encrypted ($[\![\mathbf{m}_p]\!]$) and the coordinator returns the global weights also encrypted ($[\![\mathbf{w}]\!]$). Furthermore, since the coordinator operates with the encrypted $[\![\mathbf{m}_p]\!]$ vectors, it does not need to be trusted, which is also an additional advantage over other federated models with a client–server architecture. In addition, the one-time sharing of information over the communication network (single round) makes FedHEONN more robust against attacks that attempt to violate data privacy.
- It has been formally shown that FedHEONN is more efficient, in terms of the amount of information sent over the communications network, than a traditional FL method that uses the same base learning model, depending on the number of rounds and the fraction of clients used.

Finally, FedHEONN has a number of limitations that must be taken into account:

- As it is designed for neural network without hidden layers, it has much less representation power than other deeper networks. Despite this, experimentally it has been proven to obtain competitive results, and in some scenarios, for example, IoT environments, where low computing power devices are used, this may be the only alternative for clients.
- The cost function used by the model is based on the mean squared error, since this function is what allows us to carry out the necessary mathematical formulation to analytically obtain the optimal solution. This function is specially designed for regression tasks, so in some classification tasks the model may have a slightly lower performance (as shown in Section 4) than similar ones that use cost functions more oriented to classification tasks, such as cross-entropy.
- It must be applied in a client–server architecture to prevent another client from decrypting the information, since the private key employed in an HE scheme is unique for all the clients. If encryption of data is not required, then the model could be applied in a peer-to-peer architecture.

Future work includes the design of a more powerful algorithm that uses this learning method as a building block, incorporating the advantages of HE in the training of more complex models.

CRediT authorship contribution statement

Oscar Fontenla-Romero: Conceptualization, Methodology, Investigation, Software, Writing – original draft. **Bertha Guijarro-Berdiñas:** Methodology, Formal analysis, Supervision, Writing – original draft. **Elena Hernández-Pereira:** Investigation, Software, Visualization, Validation, Writing – original draft. **Beatriz Pérez-Sánchez:** Software, Validation, Writing – original draft.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Oscar Fontenla-Romero reports financial support was provided by Fundación BBVA. Oscar Fontenla-Romero reports financial support was provided by National Plan for Scientfic and Technical Research and Innovation of the Spanish Government. Oscar Fontenla-Romero reports financial support was provided by Xunta de Galicia. Bertha Guijarro-Berdinas reports financial support was provided by Fundación BBVA. Bertha Guijarro-Berdinas reports financial support was provided by National Plan for Scientfic and Technical Research and Innovation of the Spanish Government. Bertha Guijarro-Berdinas reports financial support was provided by Xunta de Galicia. Elena Hernandez-Pereira reports financial support was provided by Fundación BBVA. Elena Hernandez-Pereira reports financial support was provided by Xunta de Galicia. Beatriz Perez-Sanchez reports financial support was provided by Fundación BBVA. Beatriz Perez-Sanchez reports financial support was provided by National Plan for Scientfic and Technical Research and Innovation of the Spanish Government. Beatriz Perez-Sanchez reports financial support was provided by Xunta de Galicia.

Data availability

Data is public and accessible.

Acknowledgments

This work has been supported by the grant Machine Learning on the Edge - Ayudas Fundación BBVA a Equipos de Investigación Científica 2019; the National Plan for Scientific and Technical Research and Innovation of the Spanish Government (Grants PID2019-109238GB-C2 and PID2021-128045OA-I00); and by the Xunta de Galicia (ED431C 2022/44) with the European Union ERDF funds.

CITIC, as a Research Center of the University System of Galicia, is funded by Consellería de Educación, Universidade e Formación Profesional of the Xunta de Galicia through the European Regional Development Fund (ERDF) and the Secretaría Xeral de Universidades (Ref. ED431G 2019/01).

Funding for open access charge: Universidade da Coruña/CISUG.

References

[1] R. van der Meulen, What edge computing means for infrastructure and operations leaders, 2018, https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders.

[2] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, B. Agüera y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Vol. 54, PMLR, 2017, pp. 1273–1282.

[3] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, D. Ramage, Federated learning for mobile keyboard prediction, 2018, https://arxiv.org/abs/1811.03604.

[4] S. Ramaswamy, R. Mathews, K. Rao, F. Beaufays, Federated learning for emoji prediction in a mobile keyboard, 2019, https://arxiv.org/abs/1906.04329.

[5] K. Sozinov, V. Vlassov, S. Girdzijauskas, Human activity recognition using federated learning, in: 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications, 2018, pp. 1103–1111.

[6] W. Yang, Y. Zhang, K. Ye, L. Li, C.-Z. Xu, Ffd: A federated learning based method for credit card fraud detection, in: Big Data – BigData 2019, 2019, pp. 18–32.

[7] Y.M. Saputra, D.T. Hoang, D.N. Nguyen, E. Dutkiewicz, M.D. Mueck, S. Srikanteswara, Energy demand prediction with federated learning for electric vehicle networks, in: IEEE Global Communications Conference, GLOBECOM, 2019, pp. 1–6.

[8] B. Hu, Y. Gao, L. Liu, H. Ma, Federated region-learning: An edge computing based framework for urban environment sensing, in: IEEE Global Communications Conference, GLOBECOM, 2018, pp. 1–7.

[9] Y. Qi, M.S. Hossain, J. Nie, X. Li, Privacy-preserving blockchain-based federated learning for traffic flow prediction, Future Gener. Comput. Syst. 117 (2021) 328–337.

[10] M. Nergiz, Collaborative colorectal cancer classification on highly class imbalanced data setting via federated neural style transfer based data augmentation, Traitement du Signal 39 (6) (2022) 2077–2086.

[11] M. Nergiz, Federated learning-based colorectal cancer classification by convolutional neural networks and general visual representation learning, Int. J. Imaging Syst. Technol. 33 (3) (2023) 951–964.

[12] S.R. Pfohl, A.M. Dai, K.A. Heller, Federated and differentially private learning for electronic health records, 2019, https://arxiv.org/abs/1911.05861.

[13] L. Huang, A.L. Shea, H. Qian, A. Masurkar, H. Deng, D. Liu, Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records, J. Biomed. Inform. 99 (2019) 103291.

[14] T.S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I.C. Paschalidis, W. Shi, Federated learning of predictive models from federated electronic health records, Int. J. Med. Inf. 112 (2018) 59–67.

[15] D. Verma, S. Julier, G. Cirincione, Federated AI for building AI solutions across multiple agencies, 2018, https://arxiv.org/abs/1809.10036.

[16] K. Powell, Nvidia clara federated learning to deliver AI to hospitals while protecting patient data, 2019, https://blogs.nvidia.com/blog/2019/12/01/clara-federated-learning/.

[17] 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, Found. Trends® Mach. Learn. 14 (1–2) (2021) 1–210.

[18] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–19.

[19] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.

[20] W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 2031–2063.

[21] J. Wang, Z.B. Charles, Z. Xu, G. Joshi, H.B. McMahan, B. Agüera y Arcas, M. Al-Shedivat, G. Andrew, S. Avestimehr, et al., A field guide to federated optimization, 2021, https://arxiv.org/abs/2107.06917.

[22] C. Dwork, Differential privacy: A survey of results, in: Proceedings of the 5th International Conference on Theory and Applications of Models of Computation, TAMC'08, Springer-Verlag, 2008, pp. 1–19.

[23] C. Dwork, A. Roth, The algorithmic foundations of differential privacy, Found. Trends Theor. Comput. Sci. 9 (2014) 211–407.

[24] R.C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: A client level perspective, 2017, https://arxiv.org/abs/1712.07557.

[25] D. Bogdanov, S. Laur, J. Willemson, Sharemind: A framework for fast privacy-preserving computations, in: Computer Security - ESORICS 2008, Springer Berlin Heidelberg, 2008, pp. 192–206.

[26] R.L. Rivest, L. Adleman, M.L. Dertouzos, On data banks and privacy homomorphisms, in: Foundations of Secure Computation, 1978, pp. 169–179.

[27] O.A. Wahab, A. Mourad, H. Otrok, T. Taleb, Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1342–1397.

[28] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, 2018, https://arxiv.org/abs/1806.00582.

[29] S.P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, A.T. Suresh, SCAFFOLD: Stochastic controlled averaging for federated learning, in: H.D. III, A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning, 119, ICML, 2020, pp. 5132–5143.

[30] S.P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S.J. Reddi, S.U. Stich, A.T. Suresh, Mime: Mimicking centralized stochastic algorithms in federated learning, 2020, https://arxiv.org/abs/2008.03606.

[31] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, 2009, pp. 169–178.

[32] Z. Brakerski, Fully homomorphic encryption without modulus switching from classical GapSVP, in: Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Vol. 7417, Springer-Verlag, 2012, pp. 868–886.

[33] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping, in: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, 2012, pp. 309–325.

[34] J. Fan, F. Vercauteren, Somewhat practical fully homomorphic encryption, IACR Cryptol, 2012, p. 144, ePrint Arch.

[35] J.-S. Coron, T. Lepoint, M. Tibouchi, Scale-invariant fully homomorphic encryption over the integers, in: Public-Key Cryptography – PKC 2014, 2014, pp. 311–328.

[36] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, B. Thorne, Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption, 2017, ArXiv abs/1711.10677.

[37] Y. Aono, T. Hayashi, L. Trieu Phong, L. Wang, Scalable and secure logistic regression via homomorphic encryption, in: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, CODASPY '16, 2016, pp. 142–144.

[38] W. Xie, Y. Wang, S.M. Boker, D.E. Brown, Privlogit: Efficient privacy-preserving logistic regression by tailoring numerical optimizers, 2016, https://arxiv.org/abs/1611.01170.

[39] L.T. Phong, Y. Aono, T. Hayashi, L. Wang, S. Moriai, Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (5).

[40] O. Fontenla-Romero, B. Guijarro-Berdiñas, B. Pérez-Sánchez, A. Alonso-Betanzos, A new convex objective function for the supervised learning of single-layer neural networks, Pattern Recognit. 43 (5) (2010) 1984–1992.

[41] O. Fontenla-Romero, B. Guijarro-Berdiñas, B. Pérez-Sánchez, Regularized one-layer neural networks for distributed and incremental environments, in: Advances in Computational Intelligence. IWANN 2021, in: Lecture Notes in Computer Science, vol. 12862, Springer International Publishing, 2021, pp. 343–355.

[42] G.H. Golub, C.F. Van Loan, Matrix Computations, fourth ed., The Johns Hopkins University Press, 2013.

[43] M.A. Iwen, B.W. Ong, A distributed and incremental SVD algorithm for agglomerative data analysis on large networks, SIAM J. Matrix Anal. Appl. 37 (4) (2016) 1699–1718.

[44] J.H. Cheon, A. Kim, M. Kim, Y. Song, Homomorphic encryption for arithmetic of approximate numbers, in: Advances in Cryptology – ASIACRYPT 2017, in: Lecture Notes in Computer Science, vol. 10624, Springer International Publishing, 2017, pp. 409–437.

[45] S.M. Fawaz, N. Belal, A. ElRefaey, M.W. Fakhr, A comparative study of homomorphic encryption schemes using microsoft seal, J. Phys. Conf. Ser. 2128 (1) (2021) 012021.

[46] D. Dua, C. Graff, UCI machine learning repository, 2017, http://archive.ics.uci.edu/ml.

[47] M. Salauddin Khan, T.D. Nath, M. Murad Hossain, A. Mukherjee, H. Bin Hasnath, T. Manhaz Meem, U. Khan, Comparison of multiclass classification techniques using dry bean dataset, Int. J. Cogn. Comput. Eng. 4 (2023) 6–20.

[48] M. Koklu, I.A. Ozkan, Multiclass classification of dry beans using computer vision and machine learning techniques, Comput. Electron. Agric. 174 (2020) 105507.

[49] J. Zhaia, T. Lia, X. Wanga, A cross-selection instance algorithm, J. Intell. Fuzzy Systems 30 (2016) 717–728.
[50] G. Słowiński, Dry beans classification using machine learning, in: Proceedings of the 29th International Workshop on Concurrency, Specification and Programming, Vol. 1613, CS & P'21, 2021, p. 0073.
[51] B. Pishgoo, A. Akbari Azirani, B. Raahemi, A hybrid distributed batch-stream processing approach for anomaly detection, Inform. Sci. 543 (2021) 309–327.
[52] H.A. Le Thi, H.M. Le, D.N. Phan, B. Tran, Stochastic DCA for minimizing a large sum of DC functions with application to multi-class logistic regression, Neural Netw. 132 (2020) 220–231.
[53] R. Sujatha, J.M. Chatterjee, A. Rohith, R.A. Ramadan, A sequential-based deep learning model for dry beans classification, in: 2023 International Conference on Smart Computing and Application, ICSCA, 2023, pp. 1–7.
[54] M. Azhari, A. Abarda, B. Ettaki, J. Zerouaoui, M. Dakkon, Higgs boson discovery using machine learning methods with pyspark, Procedia Comput. Sci. 170 (2020) 1141–1146.
[55] M. Azhari, A. Abarda, B. Ettaki, J. Zerouaoui, M. Dakkon, Using pyspark environment for solving a big data problem: Searching for supersymmetric particles, Int. J. Innov. Technol. Explor. Eng. 9 (7) (2020) 541–546.
[56] C. Za'in, M. Pratama, E. Pardede, Evolving large-scale data stream analytics based on scalable PANFIS, Knowl.-Based Syst. 166 (2019) 186–197.
[57] M. Juez-Gil, Á. Arnaiz-González, J.J. Rodríguez, C. López-Nozal, C. García-Osorio, Rotation forest for big data, Inf. Fusion 74 (2021) 39–49.
[58] C. Za'in, M. Pratama, E. Lughofer, M. Ferdaus, Q. Cai, M. Prasad, Big data analytics based on PANFIS MapReduce, Procedia Comput. Sci. 144 (2018) 140–152.
[59] A. Laila, Scaling Genetic Algorithms To Large Distributed Datasets (Ph.D. thesis), University of Kent, 2022.
[60] F. Liu, W. Xu, J. Lu, G. Zhang, A. Gretton, D.J. Sutherland, Learning deep kernels for non-parametric two-sample tests, in: Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 6316–6326.
[61] A. Bechini, F. Marcelloni, A. Segato, A MapReduce solution for associative classification of big data, Inform. Sci. 332 (2016) 33–55.

Oscar Fontenla-Romero received the Ph.D. in Computer Science in 2002 from University of A Coruña (Spain). He is currently Full Professor at the Faculty of Informatics of this University and member of its Laboratory of R&D in Artificial Intelligence (LIDIA-UDC) and the Center for Research in Information and Communication Technologies (CITIC- http://citic-research.org). His main research interests are in both applied and theoretical aspects of machine learning: large scale learning, deep learning, online learning, and distributed and federated learning. He has participated in more than 30 research projects funded by European, national and regional agencies, as well as in several agreements with companies; he has co-authored more than 130 publications in books, international journals and conferences in the field of Artificial Intelligence. He has also participated in evaluation activities and review of scientific articles for several various conferences and journals Finally, he has been part of the Board of Directors, as Secretary, of the Spanish Association for Artificial Intelligence (AEPIA) during the period between 09/18/2013 and 10/25/2018

Bertha Guijarro-Berdiñas received the Ph.D. in Computer Science in 1998 from University of A Coruña (Spain). She is currently Tenure Professor at the Faculty of Informatics of this University and member of its Laboratory of R&D in Artificial Intelligence (LIDIA-UDC) and the Center for Research in Information and Communication Technologies (CITIC- http://citic-research.org). Her main research interests are in both applied and theoretical aspects of machine learning (large scale learning, online learning, distributed and federated learning, efficiency, Green AI), knowledge-based systems and agent-based modeling. She has participated in more than 30 research projects funded by European, national and regional agencies, as well as in several agreements with companies; she has registered a system for fetal monitoring and diagnosis, and has co-authored more than 100 publications in books, international journals and conferences in the field of Artificial Intelligence.

Elena Hernandez-Pereira graduated in computer science from the University of A Coruña (Spain) in 1996. She had a predoctoral fellowship from 1998 to 1999 and a postdoctoral fellowship from 1999 to 2001 in the same university. In 2000, she received her Ph.D. degree working on the area of the application of Artificial Intelligent techniques to sleep apnea diagnosis. She is currently an Associate professor in the Computer Science Department, University of A Coruña. Hers main current research areas are: Machine Learning, Biomedical Signal Processing and Medical Decision Support Systems. She has published 15 articles of international scope journals, 12 of them indexed in the JCR and 2 book chapters. She has also 34 contributions to international conferences and participated in 28 research projects and 11 contracts with companies. She has participated in review activities of scientific articles.

Beatriz Pérez-Sánchez received the M.S. and Ph.D. degrees in Computer Science from the University of A Coruña, Spain, in 2005 and 2010, respectively. Since 2007, she has been teaching and researching in the area of knowledge of Computer Science and Artificial Intelligence of the University of A Coruña where she is currently an Associate Professor at Computer Science and Information Technology department. Her Ph.D dissertation received different mentions as the prize for the best scientific article of the Doctoral Consortium in the 2009 edition of the CAEPIA conference granted by Spanish Association of Artificial Intelligence (AEPIA) and the Doctorate Extraordinary Award granted by University of A Coruña. Regarding her current research interests, she is focused on theoretical aspects of machine learning such as, distributed, online and scalable learning as well as their application in various fields, such as engineering and biomedicine. As a result of her research, she has published different articles in international scientific journals all of them indexed in the JCR, several contributions to scientific conferences, most of them of an international nature and indexed in the CORE ranking, and different book chapters. In relation to the research experience, she has participated in several research projects funded through competitive public calls European, national or regional agencies as well as different agreements with relevant companies. She has also participated as reviewer of scientific articles for various conferences and different journals.