# Efficient and Secure Federated Learning with Ensemble of One-Layer Neural Networks

Abel Pampín-Rodríguez
*University of A Coruña*
*CITIC*
A Coruña, Spain
abel.pampin@udc.es

Oscar Fontenla-Romero
*University of A Coruña*
*CITIC*
A Coruña, Spain
oscar.fontenla@udc.es

Elena Hernández-Pereira
*University of A Coruña*
*CITIC*
A Coruña, Spain
elena.hernandez@udc.es

Bertha Guijarro-Berdiñas
*University of A Coruña*
*CITIC*
A Coruña, Spain
berta.guijarro@udc.es

***Abstract*—In this work, we propose a Federated Learning (FL) method combining an ensemble of one-layer neural networks whose optimal parameters can be obtained through a non-iterative procedure. Therefore, unlike most state-of-the-art methods, the collaborative global model can be obtained using a single round of communication between all clients of the federated scheme. It presents a computationally efficient and incremental batch aggregation process that suits the needs of a realistic federated scenario, simplifying the management of the federated training process. The model provides the same performance in identically and non-identically distributed data scenarios. Besides, the model implements a Fully Homomorphic Encryption (FHE) scheme to enhance robustness against privacy leaks or attacks, enabling clients to offload computational work to the coordinator, which operates entirely on encrypted data. We achieve an efficient and secure distributed model with an improved representation capacity for this type of architecture. The source code used in the study is made publicly available.**

***Index Terms*—Federated learning, green AI, edge computing, ensemble, homomorphic encryption, non-iid data**

## I. Introduction

Privacy requirements both ethically, in the case of sensitive user information, and legally, with recent regulations like GDPR (General Data Protection Regulation) [1] in Europe, calls for Machine Learning (ML) algorithms that respect and comply with such constraints, all while being able to build useful models from an ever growing disperse source of data.

Federated Learning (FL) encompass privacy-preserving distributed machine learning methods that allow the use of local sensitive information, unloading its processing from centralized data centers. To address privacy concerns, these methods prevent raw data from leaving its source, keeping the training data securely decentralized. Instead, they exchange only the parameters of the learned models. This approach makes them ideal candidates for constructing ML models from mobile/Internet of Things (IoT) devices and, broadly, edge computing use cases.

These local parameters are usually delivered to a centralized server, which is responsible for their aggregation and global update. This is the case for one of the most common optimization techniques used in federated environments, FedAVG [2]. It can be viewed as a generalization of a stochastic gradient descent (SGD) tailored to these scenarios, incorporating a mini-batch approach and explicitly adjusting the number of local training epochs per communication round between clients and the coordinator or central server. However, recent research [3] has shown that privacy leaks can occur through shared local model parameters, potentially inferring sensitive information. As a result, stronger privacy measures must be integrated into FL methods, such as the implementation of Differential Privacy (DP) or Secure Multiparty Computation (SMC) techniques – like the Fully Homomorphic Encryption (FHE) scheme implemented in the model presented in this paper – to guarantee a risk-free environment against both attackers and honest-but-curious participants.

The intrinsic characteristics of the problems faced by FL algorithms create a federated scenario in which the following issues must be considered: (1) there are no independent and identically distributed (IID) datasets, nor balanced clusters or classes among the separate local inputs of the devices involved, due to the inherent usage patterns and heterogeneity of each client; for the same reason, (2) significant differences in processing power may exist, and massive scalability is required; (3) in contrast to controlled distributed ML environments, the bandwidth and communication costs, along with client availability, pose the real bottleneck for the training process of the global federated model. These challenges highlight the need for resource-aware FL algorithms that address both data and device heterogeneity.

In this work, we provide a solution to the aforementioned problems by means of a highly efficient, parallelizable, homomorphically encrypted federated learning algorithm with incremental training capabilities and a closed-form solution. This algorithm is robust against model inversion attacks and delivers consistent results for both IID and non-IID datasets, regardless of the number of clients participating. Specifically, we present an ensemble (applying the bagging technique known as Random Patches [4]) of regularized one-layer feedforward neural networks which minimizes the Mean Squared Error (MSE) measured *before* the activation function. With this setup, we aim to leverage the strengths of these simple models and achieve performance near that of much more complex architectures.

Simple neural network models, as those used in this work, remain highly relevant today – even with the advent of increasingly large and complex neural networks – as they are often sufficient to solve many tasks, easier to interpret, train quickly, and handle large datasets efficiently, though they lack the representational power of deeper networks. Additionally, these models could serve as a building block for more powerful algorithms.

The rest of this paper is organized as follows. Section-II establishes the base model of this work. The proposed method is presented and detailed in Section-III. Experimental results are discussed in Section-IV. Finally, concluding remarks and future work are presented in Section-V.

## II. Background

Our proposal builds on previous research [5]–[7], which we will introduce in this section for clarity. Specifically, we will provide the necessary background for the base model of this work, called FedHEONN [7], and its distributed, secure application to federated scenarios. First, we will define the terminology based on the model – a one-layer feedforward neural network (i.e., no hidden layers) – with a centralized training set. Then, we will describe the federated scheme in detail.

Consider a centralized training dataset defined by an input matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, where $m$ is the number of features (including the bias), and $n$ the number of instances, with a desired output matrix defined by $\boldsymbol{d} \in \mathbb{R}^{n\times c}$, where $c$ is the number of outputs (in the remainder, we assume only one output, i.e., $\boldsymbol{d} \in \mathbb{R}^{n\times 1}$, for simplicity). The neural network parameters are defined by a weight vector (including the bias), $\boldsymbol{w} \in \mathbb{R}^{m\times 1}$, and, consequently, the output of the model is $\boldsymbol{y} = f(\boldsymbol{X}^T\boldsymbol{w})$, where $f : \mathbb{R} \to \mathbb{R}$ is the nonlinear activation function at the output neuron.

Usually, the optimal weights in neural networks are obtained through an iterative process that minimizes a cost function, with the mean squared error (MSE) being one of the most commonly used for supervised learning. This cost function compares the actual output $\boldsymbol{y}$ of the network to the desired output $\boldsymbol{d}$. However, previous work [5] has shown that an alternative is to minimize the MSE measured *before* the activation function – i.e., between $\boldsymbol{X}^T\boldsymbol{w}$ and $\bar{\boldsymbol{d}} = f^{-1}(\boldsymbol{d})$ – resulting in the following cost function, which includes a regularization term based on the L2 norm to avoid overfitting:

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

where $\boldsymbol{F} = \text{diag}\big(f'(\bar{d}_1), \ldots, f'(\bar{d}_n)\big)$ is a diagonal matrix formed by the derivatives of the $f$ function for the components of $\bar{\boldsymbol{d}}$, and $\lambda$ is a positive scalar hyperparameter that controls the magnitude of the penalty term in the regularization.

Despite using nonlinear activation functions, the cost function in Eq.(1) has the advantage of being convex, allowing the global optimum to be obtained directly through a closed-form solution for any given $\boldsymbol{X}$ and $\boldsymbol{d}$. Specifically, its minimum is found by equating its derivative to zero, which results in the following system of linear equations, solving for a $\boldsymbol{w}$ that satisfies:

$$(\boldsymbol{XFFX}^T + \lambda\boldsymbol{I})\,\boldsymbol{w} = \boldsymbol{XF F}\bar{\boldsymbol{d}} \quad (2)$$

The most costly operations of the system of linear equations in Eq.(2) involve matrix $(\boldsymbol{XFFX}^T + \lambda\boldsymbol{I})$ and its inverse, with computational complexities of $O(m^2n)$ and $O(m^3)$, respectively. This implies high efficiency for small $m$ feature datasets, but it quickly becomes computationally demanding for larger ones. To obtain the solution $\boldsymbol{w}$ efficiently, the previous system of linear equations is transformed [6] using Singular Value Decomposition (SVD). This is achieved by factorizing matrix $\boldsymbol{XF} = \boldsymbol{USV}^T$, where $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ is a diagonal matrix with $r$ non-zero singular values, with $r = \text{rank}(\boldsymbol{XF}) \le \min(m, n)$. With this approach, the system of equations changes to

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

resulting in an optimal solution that directly provides the minimum error for a given training set by:

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

Using an economy-sized decomposition that retains all the relevant information, the effective dimensions of matrices $\boldsymbol{U}$ and $\boldsymbol{S}$ become $m \times r$ and $r \times r$, respectively, lowering the computational complexity to $O(mnr)$. In addition, since $(\boldsymbol{SS}^T + \lambda\boldsymbol{I})$ is a diagonal matrix, its inverse reduces complexity to $O(r)$, making for an efficient solution whether working with $m \gg n$ or $m \ll n$ datasets.

The application of the preceding method to a federated scenario, specifically to a type of FL known as Horizontal Federated Learning (HFL), where the dataset is partitioned into $P$ submatrices $\boldsymbol{X} = [\boldsymbol{X}_1 \mid \cdots \mid \boldsymbol{X}_P]$ distributed among the clients – such that they share the same features but differ in the number of samples – requires adopting the following measures to reformulate the previous research and achieve the solution presented in Eq.(4):

- *Incremental SVD computation*: The term $\boldsymbol{U}(\boldsymbol{SS}^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{U}^T$ involves global matrices $\boldsymbol{U}$ and $\boldsymbol{S}$, which were originally computed from the centralized SVD of $\boldsymbol{XF}$. In a HFL scenario, they can be computed incrementally and distributed across clients, as shown by Iwen and Ong [8]. In their work, they demonstrated that a matrix decomposed into $P$ partitions $\boldsymbol{A} = [\boldsymbol{A}_1 \mid \cdots \mid \boldsymbol{A}_P]$ shares the same singular values $\boldsymbol{S}$ and left singular vectors $\boldsymbol{U}$ as a matrix $\boldsymbol{B} = [\boldsymbol{U}_1\boldsymbol{S}_1 \mid \cdots \mid \boldsymbol{U}_P\boldsymbol{S}_P]$, where $\text{SVD}(\boldsymbol{A}_p) = \boldsymbol{U}_p\boldsymbol{S}_p\boldsymbol{V}_p \quad \forall p \in \{1, \ldots, P\}$. Furthermore, this partial SVD merging scheme maintains numerical robustness even in the presence of data corruption, rounding errors and reduced rank. Therefore, for our purposes, we compute the economy-sized SVD of the global matrix $\boldsymbol{XF}$ using partial SVDs calculated across all clients, as follows:

$$\begin{aligned}\mathrm{SVD}(\boldsymbol{XF}) &= \mathrm{SVD}\big([\boldsymbol{X}_1\boldsymbol{F}_1 \mid \cdots \mid \boldsymbol{X}_P\boldsymbol{F}_P]\big)\\ &= \mathrm{SVD}\big([\boldsymbol{U}_1\boldsymbol{S}_1 \mid \cdots \mid \boldsymbol{U}_P\boldsymbol{S}_P]\big)\end{aligned} \tag{5}$$

- *Incremental block partitioned matrix product*: The global term $\boldsymbol{XFF\bar{d}}$, referred to as $\boldsymbol{m} \in \mathbb{R}^{m\times 1}$ from now on, involves only algebra within the submatrices of the factors, so it can be viewed as a block-partitioned matrix product. This can be partially and incrementally computed using the local information provided by $P$ clients:

$$\boldsymbol{m} = \boldsymbol{XFF\bar{d}} = [\boldsymbol{X}_1 \mid \cdots \mid \boldsymbol{X}_P]\begin{bmatrix}\boldsymbol{F}_1\\ \vdots\\ \boldsymbol{F}_P\end{bmatrix}\begin{bmatrix}\boldsymbol{F}_1\\ \vdots\\ \boldsymbol{F}_P\end{bmatrix}\begin{bmatrix}\bar{\boldsymbol{d}}_1\\ \vdots\\ \bar{\boldsymbol{d}}_P\end{bmatrix}$$

$$\boldsymbol{m} = \sum_{p=1}^{P}\boldsymbol{X}_p\boldsymbol{F}_p\boldsymbol{F}_p\bar{\boldsymbol{d}}_p \tag{6}$$

Specifically, suppose that at client $p$, we have a local dataset of $n_p$ examples, $\boldsymbol{X}_p \in \mathbb{R}^{m\times n_p}$, and that $\boldsymbol{m}_p = \boldsymbol{X}_p\boldsymbol{F}_p\boldsymbol{F}_p\bar{\boldsymbol{d}}_p$ is computed. When new data $\boldsymbol{X}_k$ becomes available – either from another client $k$ or from a new time window – the new vector $\boldsymbol{m}_{p|k}$, which combines the information from both $\boldsymbol{X}_p$ and $\boldsymbol{X}_k$, can be easily obtained by adding the term dependent on the new data block, $\boldsymbol{m}_k$, to $\boldsymbol{m}_p$.

$$\boldsymbol{m}_{p|k} = \boldsymbol{m}_p + \boldsymbol{X}_k\boldsymbol{F}_k\boldsymbol{F}_k\bar{\boldsymbol{d}}_k \tag{7}$$

- *Enforced security*: Although this method does not transmit raw data across the network – only locally computed matrices $\boldsymbol{U}_p\boldsymbol{S}_p$ and $\boldsymbol{m}_p$ – adhering to the privacy-by-design principle of FL algorithms, a FHE mechanism is deployed as an extra layer of security against malicious attacks that may infer local data used during the training process, as demonstrated in recent works [9]. This Homomorphic Encryption (HE) scheme allows operations to be performed on encrypted data, yielding the same result as if the operation were executed using plain data. We apply this encryption to the $\boldsymbol{m}_p$ vector – sent by each client to the coordinator responsible for aggregation – since this information may be more vulnerable to attacks compared to the partial matrix $\boldsymbol{U}_p\boldsymbol{S}_p$ from the SVD factorization of $\boldsymbol{X}_p\boldsymbol{F}_p$.

$$\begin{aligned}\boldsymbol{m} = \boldsymbol{XFF\bar{d}} &= \sum_{p=1}^{P}\boldsymbol{X}_p\boldsymbol{F}_p\boldsymbol{F}_p\bar{\boldsymbol{d}}_p\\ [\![\, \boldsymbol{m} \,]\!] &= \sum_{p=1}^{P}[\![\, \boldsymbol{m}_p \,]\!]\end{aligned} \tag{8}$$

where $[\![\cdot]\!]$ denotes the HE operator. In this paper, we use a CKKS HE scheme [10] due to its ability to support approximate arithmetic over real numbers, facilitated by homomorphic addition and multiplication. These two operations are sufficient to implement the proposed method.

Regarding the last point, note that the CKKS implementation imposes no limit on chained addition operations (affecting $P$ times $\boldsymbol{m}$). However, it does restrict multiplications to no more than two sequential operations before losses occur [11]. This limitation does not affect our algorithm, as only two multiplications are needed to compute the solution $\boldsymbol{w}$ in Eq.(4), regardless of the number of clients $P$.

## III. Proposed method

Our proposal introduces a federated learning method that generates an ensemble of one-layer neural networks, built upon the FedHEONN model, balancing an improvement in its base performance with a slight increase in its overall execution time.

Each client trains multiple base estimators in parallel using different subsets of data, by means of a random feature and sample selection technique that needs to be properly synchronized. Later on, the aggregation of local model parameters coming from different clients is done efficiently and securely, retaining the scalability, privacy and good performance of the original model regardless of the number of clients and non-IID datasets. The main steps of the proposed method are described below and summarized in Fig.1:

1) *Setup*: A preliminary communication round is conducted, during which cryptographic keys are exchanged, and feature synchronization takes place.
2) *Local training*: Each of the $P$ clients trains $T$ estimators using $r$ random patches of their local input data $\boldsymbol{X}_p$, producing $T$ encrypted matrices $[\![\boldsymbol{m}_p]\!], \boldsymbol{U}_p$, and $\boldsymbol{S}_p$.
3) *Sending parameters*: During a second communication round, clients upload their locally computed parameters to the coordinator for its queued secure aggregation.
4) *Computation of global optimal parameters*: The coordinator incrementally processes the received data and can compute the optimal parameters at any time.
5) *Receiving optimal parameters and inference*: In the final communication round, the clients receive a list of $T$ encrypted vectors $[\![\boldsymbol{w}]\!]$, used for average or majority vote predictions.

In this section we will first present the ensemble technique, called Random Patches. Then we will explain how we detached the incremental aggregation from the computation of the global optimal weights of the base algorithm and, finally, how we integrated both solutions into the proposed method.

### A. Bagging with Random Patches

Introduced by Gilles Louppe et al. [4], Random Patches (RP) serves as an encompassing generalization of bootstrap aggregating, offering a simple yet effective ensemble framework particularly suited in contexts involving Big Data, distributed databases, and embedded systems. This technique builds each individual estimator from a random patch of data, obtained by selecting random subsets of *both* instances and features from the entire dataset. RP claims comparable accuracy to other popular ensemble methods, while significantly lowering the memory requirements.

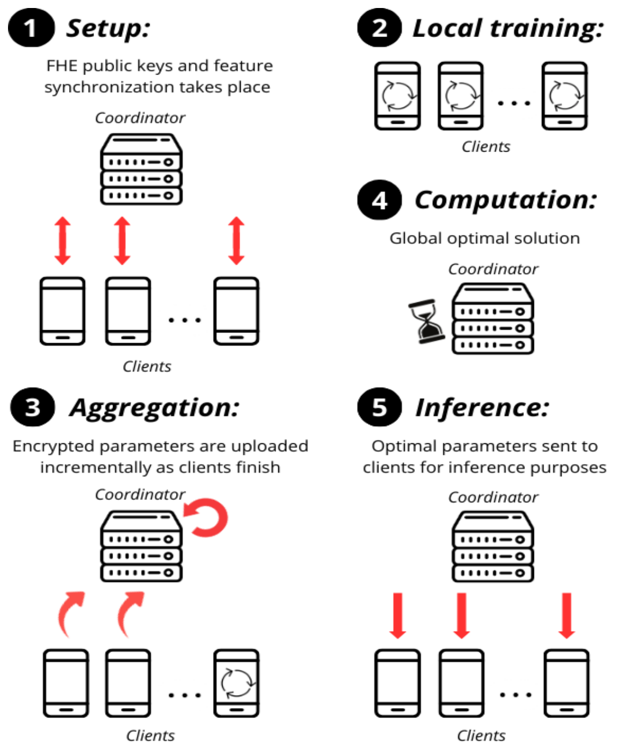
1 Setup:
FHE public keys and feature synchronization takes place
Coordinator
Clients
2 Local training:
Clients
4 Computation:
Global optimal solution
Coordinator
3 Aggregation:
Encrypted parameters are uploaded incrementally as clients finish
Coordinator
Clients
5 Inference:
Optimal parameters sent to clients for inference purposes
Coordinator
Clients

Fig. 1. Main steps of the proposed method.

This wrapper ensemble method can be described as follows: Let $R(\rho_s, \rho_f, D)$ be the set of all random patches of size $\rho_s N_s \times \rho_f N_f$ that can be drawn from dataset $D$, where $N_s$ and $N_f$ are the number of samples and features in $D$, and $\rho_s$, $\rho_f \in [0, 1]$ are hyperparameters that control the fraction of samples and features in a patch; i.e., $R(\rho_s, \rho_f, D)$ is the set of all possible subsets of $\rho_s N_s$ samples with $\rho_f N_f$ features. This method works as follows:

1) Draw a patch $r \sim U(R(\rho_s, \rho_f, D))$ at random.
2) Train a base estimator on the selected patch $r$.
3) Repeat steps 1-2 for a preassigned number $T$ of estimators.
4) Aggregate the predictions of the $T$ base estimators through averaging or majority voting.

The Random Patches method generalizes both the Pasting Votes (PV) [12] and the Random Subspace (RS) [13] algorithms, as these are merely special cases of RP. Specifically, setting $\rho_s = 1$ reduces RP to RS, while setting $\rho_f = 1$ yields PV, making RP a comprehensive combination of bagging and subspace ensembles techniques.

As previously mentioned, RP offers a significant advantage in contexts where the dataset size far exceeds the available memory. By appropriately sizing the random patches, RP can produce an ensemble model that performs as effectively as if the entire dataset had been used directly. This makes it particularly valuable in environments constrained by limited memory, expensive data access or embedded systems where a federated application is of interest.

Formally, consider a situation where the memory required for a data unit is quantified as $M$, and the available memory is constrained by a given threshold $M_{\text{max}}$, representing the maximum data units that can be loaded to train a single estimator (naively disregarding on-line algorithms and the memory inherently consumed by the estimator itself). Then, since the memory required by a random patch would be $(\rho_s N_s)(\rho_f N_f)$, constraining memory by $M_{\text{max}}$ is equivalent to limiting the relative patch size to:

$$\rho_s \rho_f \leq \frac{M_{\text{max}}}{N_s N_f} \tag{9}$$

Like other bagging methods, Random Patches is naturally parallelizable, as the extraction and training of each random patch for every estimator can be performed simultaneously and independently, before aggregating or averaging their predictions.

In summary, this technique introduces three key hyperparameters ($T$, $\rho_s$, $\rho_f$) to control the number of estimators and the fraction of samples and features in a patch, alongside the regularization parameter ($\lambda$) inherited from the base model. Additionally, two optional flags ($\beta_s$, $\beta_f$) enable resampling with replacement for samples and features, offering further flexibility.

## B. *Incremental batch training process*

Originally, the aggregation of client information was performed once globally before computing the optimal solution, all within the same task. We decoupled aggregation from solution computation, introducing a novel distinction from the base model [7].

This approach enables computing the solution at any point during the progressive incorporation of local data from the $P$ clients. This eliminates the need to wait for all client matrices to be processed (a bottleneck caused by the slowest client) and, at the same time, allows batches or small groups of clients to submit their model data (or even submit them individually), as soon as their respective local training is completed and available. These contributions can then be promptly integrated into the global matrices $\boldsymbol{U}$, $\boldsymbol{S}$ and $\boldsymbol{m}$, enhancing scalability and adaptability in federated scenarios.

To illustrate this process, consider $P$ clients (respective partitions of the dataset $\boldsymbol{X}$) organized into $K$ groups, where each group contains a disjoint subset of clients. Let $\mathcal{I}_k$ denote the set of indices corresponding to the clients in group $k$. The global dataset can then be expressed as $\boldsymbol{X} = [\boldsymbol{X}^1 \mid \cdots \mid \boldsymbol{X}^K]$, where $\boldsymbol{X}^k = \cup \boldsymbol{X}_p \ \ \forall p \in \mathcal{I}_k$. Obviously, $K \leq P$ is always satisfied, with equality occurring only when each group comprises a single client, although it is more common to have $K \ll P$. This batch processing framework aggregates local matrices from each client into group-level matrices $\boldsymbol{U}^k$, $\boldsymbol{S}^k$ and $\boldsymbol{m}^k$, which are then used to update the global model incrementally, offering a more flexible and scalable training process.

Empirically, we have successfully validated that this clustering approach is as effective as the standard distributed method described in Eq.(5). Specifically, we checked that the singular value decomposition (SVD) of the global matrix $\boldsymbol{XF}$ remains equivalent across hierarchical levels of aggregation.

## *C. Ensemble of FedHEONN*

The implementation of the proposed method in the HFL scenario requires a *feature synchronization* process to coordinate the randomly selected features, $m_f = \lfloor \rho_f N_f \rfloor$, used across the $P$ different clients for sampling the $\boldsymbol{r}$-$i$th patch of local data. This synchronization is essential because correct aggregation of the ensemble of matrices $\{[\![\boldsymbol{m}_p]\!]\}$ and $\{\boldsymbol{US}_p\}$ between clients cannot be achieved if their local input matrices $\boldsymbol{X}_p \in \mathbb{R}^{(m_f)\times(\rho_s n_p)}$ result from mismatched subsets of features.

To address this, the coordinator (or central server) generates and distributes lists of randomly selected attributes to all clients during a communication round prior to training. This process is handled by a function that takes as input the number of expected estimators $T$, the desired fraction of features $\rho_f$, and the total number of features $N_f$; in order to construct $T$ independent lists, each containing $m_f$ random attributes.

We now outline the key algorithms for both entities in the federated framework: algorithms 1 and 2 describe the client's fit and predict operations, respectively. Algorithm 3 details the coordinator's process for incrementally aggregating data from a certain group $k$ of clients, while algorithm 4 describes how the global optimal weights are computed.

Combining the base model and the ensemble method primarily involved wrapping its main operations so that they could be performed $T$ times, with each instance corresponding to a random patch $r$ extracted from the local dataset of each client. These core processes are framed within boxes for each algorithm and correspond to the original unaltered operation of the FedHEONN model. The proposed method wraps these operations and parallelizes them.

Thus, for instance, lines 19-27 of algorithm 1 represent the client's fit training for a certain local data matrix $\boldsymbol{X}_p$ and true output vector $\boldsymbol{d}_p$, while lines 8-18 of algorithm 3 reproduces the bulk of the coordinator aggregation process, in guidance with the incremental SVD computation and block partitioned matrix product presented in Section-II. The solution in Eq.(4) is computed by the coordinator at any time in lines 10-11 of algorithm 4 and prediction at client $p$ comes from applying lines 12-13 of algorithm 2 for any local input test matrix $X^{test}$, upon receiving the weights.

## IV. Results

In this section, we evaluate the performance of the proposed method in terms of accuracy. The experimental study was conducted on a computer with an NVidia GTX 1660 Ti GPU, an Intel i7-9750H 2.6 GHz 6-core CPU, and 16 GB of RAM. The code used in this study is available at: https://github.com/abeludc93/fedheonn_ensemble. The proposed method was programmed entirely in Python, utilizing *Numpy* and *TenSEAL* as core libraries for matrices calculations and homomorphic encryption operations on vectors, respectively.

**Algorithm 1** FedHEONN-ensemble **client fit**

```
Input:
 1: $\boldsymbol{X}_p \in \mathbb{R}^{m\times n_p}$                    ▷ Local data matrix
 2: $\boldsymbol{d}_p \in \mathbb{R}^{n_p\times 1}$                   ▷ True output vector
 3: $f$                                        ▷ Act. function
 4: $T \in \mathbb{N}$                              ▷ No. of $T$ estimators
 5: $\rho_s \in (0,1]$                     ▷ Fraction of inst. in rand. patch $r$
 6: $\beta_s :=$ true or false              ▷ Inst. sampling w/ replacement
 7: $\rho_f \in (0,1]$                     ▷ Fraction of feat. in rand. patch $r$
 8: $\beta_f :=$ true or false              ▷ Feat. sampling w/ replacement
Output:
 9: $\{[\![\boldsymbol{m}_p^i]\!]\}_{i=1}^T,\ \{\boldsymbol{US}_p^i\}_{i=1}^T$          ▷ Lists of $T$ $\boldsymbol{m}_p$ and $\boldsymbol{US}_p$

10: function CLIENT:ENS_FIT($\boldsymbol{X}_p, \boldsymbol{d}_p, f, T, \rho_s, \beta_s, \rho_f, \beta_f$):
11:     $\{\mathcal{I}_f\}_{i=1}^T \leftarrow$ COORD:RAND_FEATURES($T, \rho_f, \beta_f, m$)
12:     ▷ Lists of $T$ rand. feat. indices, generated and shared by
          the coordinator                                        ◁
13:     for all $i \in \{1,\dots,T\}$ do
14:         $\mathcal{I}_f \leftarrow \{\mathcal{I}_f^i\}$            ▷ $i$-th series of rand. feat. indices
15:         $\mathcal{I}_s \leftarrow$ RAND(array=$n_p$, size=$\rho_s \times n_p$, replace=$\beta_s$)
16:         ▷ Rand. inst. sampling                              ◁
17:         $\boldsymbol{X}_p^i \leftarrow \boldsymbol{X}_p[\mathcal{I}_f, \mathcal{I}_s]$              ▷ Index data matrix patch
18:         $\boldsymbol{d}_p^i \leftarrow \boldsymbol{d}_p[\mathcal{I}_f, \mathcal{I}_s]$             ▷ Index output vector patch
19:         ▷            Fit estimator pairs $\boldsymbol{X}_p^i, \boldsymbol{d}_p^i$              ◁
20:         $\boldsymbol{X}_p^i \leftarrow$ [ones(1, $n_p$); $\boldsymbol{X}_p^i$]                 ▷ Added bias
21:         $\bar{\boldsymbol{d}}_p \leftarrow f^{-1}(\boldsymbol{d}_p^i)$                ▷ Act. function inverse
22:         $\boldsymbol{f}_p \leftarrow f'(\bar{\boldsymbol{d}}_p)$               ▷ Act. function derivative
23:         $\boldsymbol{F}_p \leftarrow$ diag($\boldsymbol{f}_p$)                      ▷ Diagonal matrix
24:         $[\boldsymbol{U}_p^i, \boldsymbol{S}_p^i, \sim] \leftarrow$ SVD($\boldsymbol{X}_p^i \boldsymbol{F}_p$)          ▷ Economy SVD
25:         $\boldsymbol{US}_p^i \leftarrow \boldsymbol{U}_p^i$ diag($\boldsymbol{S}_p^i$)               ▷ Matrix product
26:         $\boldsymbol{m}_p^i \leftarrow \boldsymbol{X}_p^i(\boldsymbol{f}_p \circ \boldsymbol{f}_p \circ \bar{\boldsymbol{d}}_p^i)$              ▷ Vector $\boldsymbol{m}_p^i$
27:         $[\![\boldsymbol{m}_p^i]\!] \leftarrow$ ckks_encryption($\boldsymbol{m}_p^i$)       ▷ FHE encryption
        return $\{[\![\boldsymbol{m}_p^i]\!]\}_{i=1}^T, \{\boldsymbol{US}_p^i\}_{i=1}^T$
28:     ▷ Lists of $T$ encrypted vectors $[\![\boldsymbol{m}_p]\!]$ and matrices $\boldsymbol{US}_p$
          fitted for client data matrix $\boldsymbol{X}_p$                 ◁
```

**Algorithm 2** FedHEONN-ensemble **client prediction**

```
Input:
 1: $\boldsymbol{X}_p^{test} \in \mathbb{R}^{m\times n_p}$                    ▷ Local test matrix
 2: $[\![\boldsymbol{w}]\!] \in \mathbb{R}^{m\times 1}$                ▷ Encrypted opt. weights
 3: $f$                                        ▷ Act. function
 4: $T \in \mathbb{N}$                                 ▷ No. of estimators
 5: $\{\mathcal{I}_f\}_{i=1}^T$            ▷ Rand. feat. indices used during training
Output:
 6: $\boldsymbol{y} \in \mathbb{R}^{n_p\times 1}$                ▷ Prediction vector on test

 7: function CLIENT:ENS_PRED($\boldsymbol{X}_p^{test}, [\![\boldsymbol{w}]\!], f, T, \{\mathcal{I}_f\}_{i=1}^T$):
 8:     $\boldsymbol{w} \leftarrow$ ckks_decryption($[\![\boldsymbol{w}]\!]$)          ▷ Decrypt opt. weights
 9:     for all $i$ in $T$ do
10:         $\mathcal{I}_f \leftarrow \{\mathcal{I}_f^i\}$            ▷ $i$-th series of rand. feat. indices
11:         $\boldsymbol{X}_p \leftarrow \boldsymbol{X}_p^{test}[\mathcal{I}_f, :]$             ▷ Index feat. from matrix
12:         ▷          Prediction for $i$-th estimator             ◁
13:         $\boldsymbol{y}_i \leftarrow f(\boldsymbol{X}_p^T \boldsymbol{w})$           ▷ $f$ is applied component-wise
14:     if CLASSIFICATION then
15:         $\boldsymbol{y}^{test} \leftarrow \arg\max[\boldsymbol{y}_1 \mid \cdots \mid \boldsymbol{y}_T]$            ▷ Majority vote
16:     else if REGRESSION then
17:         $\boldsymbol{y}^{test} \leftarrow \frac{1}{T}\sum_{i=1}^T \boldsymbol{y}_i$                           ▷ Average
        return $\boldsymbol{y}^{test}$                              ▷ Prediction vector
```

**Algorithm 3** FedHEONN-ensemble **coordinator partial agg.**

```
Input:
1: $\{M^k\}$, $\{US^k\}$
2: ▷ Lists representing $\{[\![m_p^i]\!]\}_{i=1}^T, \{[\![US_p^i]\!]\}_{i=1}^T \; \forall p \in \mathcal{I}_k$ ◁

3: function COORD:ENS_PARTIAL_AGG($\{M^k\}, \{US^k\}$):
4:     $T \leftarrow$ LENGTH($\{M_1^k\}$)    ▷ Infer no. of estimators $T$
5:     for all $i$ in $T$ do
6:         $M^k \leftarrow \{M_i^k\}$    ▷ Select $i$-th series
7:         $US^k \leftarrow \{US_i^k\}$
8:         ▷ Partial aggregation of $M^k, US^k$ ◁
9:         if previous $[\![m]\!], U, S$ are available then
10:            $[\![m]\!] \leftarrow$ Stored $[\![m]\!]$    ▷ Init. vector $[\![m]\!]$
11:            $U, S \leftarrow$ Stored $U, S$    ▷ Init. matrices $U, S$
12:        else
13:            $[\![m]\!] \leftarrow \mathbf{0}$    ▷ Zero vector
14:            $U, S \leftarrow [\,], [\,]$    ▷ Empty matrices
15:        for all $[\![m_p]\!], US_p$ in $(M^k, US^k)$ do
16:            ▷ Incremental SVD and FHE addition ◁
17:            $[U, S, \sim] \leftarrow$ SVD$([US \mid US_p])$
18:            $[\![m]\!] \leftarrow [\![m]\!] + [\![m_p]\!]$
19:        Save $[\![m]\!]$, $U$, $S$    ▷ Future agg. of new information
```

**Algorithm 4** FedHEONN-ensemble **coordinator opt. weights**

```
Input:
1: $\{[\![m]\!]\}, \{U\}, \{S\}$
2: ▷ Saved lists of $T$ global matrices $U, S$ and $[\![m]\!]$ ◁
Output:
3: $\{[\![w]\!]\}_{i=1}^T$    ▷ Encrypted list of opt. weights vectors

4: function COORD:ENS_WEIGHTS($\{[\![m]\!]\}, \{U\}, \{S\}$):
5:     $T \leftarrow$ LENGTH($\{[\![m]\!]_1\}$)    ▷ Infer no. of estimators $T$
6:     for all $i$ in $T$ do
7:         ▷ Index $i$-th global matrices $U, S$ and $[\![m]\!]$ ◁
8:         $[\![m]\!] \leftarrow \{[\![m]\!]_i\}$
9:         $U, S \leftarrow \{U_i\}, \{S_i\}$
10:        ▷ Calculate opt. weights for i-th estimator ◁
11:        $[\![w_i]\!] \leftarrow U\,(SS^T + \lambda I)^{-1}\,(U^T\,[\![m]\!])$    ▷ Solution
       return $\{[\![w_1]\!], \ldots, [\![w_T]\!]\}$
```

### A. Datasets

The experimental methodology involved standardizing features and performing a 10-fold cross-validation within a grid-search hyperparameter space, tuned for optimal accuracy results. This process was carried out for both the original model, where $\lambda$ is the only hyperparameter, and the ensemble method, which involves hyperparameters $\lambda, \rho_s, \rho_f, T, \beta_s, \beta_f$. Both IID (random shuffling) and non-IID (nonhomogeneous distribution of the sorted dataset) setups were considered, together with a varying size of the number of clients $P$ in the federated network. Since the results remained equivalent across both scenarios, only the IID case is reported. Table I presents the main characteristics of the classification datasets used in this study.

TABLE I
CHARACTERISTICS OF THE DATASETS

| Dataset | Samples | Attributes | Classes |
|---|---|---|---|
| Obesity [14] | 2,111 | 16 | 7 |
| Hand-written digits (ORHD) [15] | 5,620 | 64 | 10 |
| DryBean [16] | 13,611 | 16 | 7 |
| MiniBoone [17] | 130,065 | 50 | 2 |
| Skin [18] | 245,057 | 3 | 2 |

### B. Performance comparison of the proposed method with the base model

This experiment evaluates the effectiveness of the Random Patches ensemble method compared to the base model. A grid search cross-validation was performed, aimed to both analyze the sensitivity of the hyperparameters $(T, \rho_s, \rho_f, \beta_s, \beta_f)$ and identify the best achievable accuracy relative to the base model. The results (mean accuracy and standard deviation) are provided in Table II. As can be seen, compared to the base method, the proposed method obtains better results in all cases. It yielded significant improvements in accuracy in some cases, such as *Obesity* (6.8% increase) and *ORHD* (1.1% increase), while others showed modest gains. Additionally, the optimal hyperparameters for the proposed method varied by dataset, and higher numbers of base estimators $T$ did not always result in better performance, emphasizing the need for targeted smart hyperparameter tuning. Besides, we have included, as a reference for each dataset, the performance of a centralized (non-federated) learning model available in the scientific literature. As can be seen, the performance of the proposed method is very competitive considering that it is an algorithm that proposes distributed learning and uses homomorphic encryption.

### C. Parallelizing the proposed method

We also conducted a study to evaluate time performances achieved by parallelizing training tasks (clients) and partial aggregation/optimal weight computation (coordinator). The experiment was carried out using the *ORHD* dataset, varying the relative patch sizes $(\rho_s \times \rho_f)$ and the number of base estimators $T$, for two different client counts $P = \{4, 16\}$. Figure 2 illustrates the execution times for each scenario in both parallel (solid) and serial (dotted) configurations for an increasing number of assembled estimators.

Key findings indicate that parallelization proved effective for client-side training and coordinator-side partial aggregation tasks. These tasks involve relatively lightweight operations, such as SVD decompositions on plain data (both tasks) and homomorphic additions on encrypted vectors (partial aggregation only). However, parallelizing the computation of optimal weights at the coordinator proved inefficient for alleviating the high computational cost of homomorphic matrix-vector multiplication, due to the overhead of serializing encrypted vectors and managing process pools. This added cost negated any gains, except in cases with a high number of base estimators $T$ or large relative patch sizes $(\rho_s \times \rho_f)$.

Overall, client-side parallelization is consistently beneficial across all configurations of $T$ and relative patch sizes, benefiting computationally limited multi-core edge devices.

TABLE II
CROSS-VALIDATION RESULTS: MEAN ACCURACY FOR THE BASE MODEL AND THE PROPOSED ENSEMBLE VARIANT
COMPARISON WITH CENTRALIZED CLASSICAL METHODS: SVM, K-NN, C-NN AND VFDT

| Dataset | Method | Hyperparameters | | | | | | | Accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\lambda$ | Activation f. | $T$ | $\rho_s$ | $\beta_s$ | $\rho_f$ | $\beta_f$ | $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ | |
| Obesity | Base | 0.1 | *relu* | - | - | - | - | - | 74.61 % | 4.04 % |
| | Proposed | 0.01 | *relu* | 50 | 0.4 | × | 0.8 | × | 81.38 % | 1.72 % |
| | SVM [19] | | | | | | | | 87.47 % | |
| ORHD | Base | 0.1 | *logs* | - | - | - | - | - | 93.39 % | 1.69 % |
| | Proposed | 0.001 | *logs* | 96 | 0.1 | × | 0.75 | × | 94.51 % | 1.50 % |
| | SVM [15] | | | | | | | | 97.61 % | |
| DryBean | Base | 0.001 | *logs* | - | - | - | - | - | 90.07 % | 1.08 % |
| | Proposed | 0.001 | *logs* | 2 | 0.5 | ✓ | 0.9 | × | 90.61 % | 1.05 % |
| | k-NN [20] | | | | | | | | 89.00 % | |
| MiniBoone | Base | 0.0001 | *logs* | - | - | - | - | - | 89.90 % | 0.18 % |
| | Proposed | 0.01 | *logs* | 2 | 1 | × | 0.7 | × | 90.02 % | 0.23 % |
| | Condensed NN [21] | | | | | | | | 82.96 % | |
| Skin | Base | 0 | *logs* | - | - | - | - | - | 92.51 % | 0.18 % |
| | Proposed | 0 | *logs* | 2 | 0.05 | × | 1 | × | 92.54 % | 0.21 % |
| | Very Fast Decision Tree [22] | | | | | | | | 93.33 % | |

Regarding different client configurations, aggregation time increases with the number of clients, requiring more operations in order to aggregate global matrices $\boldsymbol{U}$, $\boldsymbol{S}$ and $[\![\boldsymbol{m}]\!]$. However, this task benefits significantly from parallelization, highlighting the scalability of the proposed algorithm.

Client fit time decreases as the dataset becomes more partitioned. In contrast, solution computation remains unaffected by the number of clients, as it operates solely on the previously aggregated global matrices.

## V. CONCLUSIONS

In this paper we presented an ensemble method, Random Patches, to a secure FL base model, FedHEONN, with a closed-form solution and incremental training nature, enhancing its representational capacity and providing additional reasons to use it in simple applications, despite being based on a single-layer feedforward neural network.

While the performance of the ensemble method depends heavily on the dataset, notable gains were observed that do not depend exclusively on a high number of base estimators $T$, but rather on the variance introduced by combining random patches of data.

To reduce overall training times with the proposed ensemble method, we parallelized the training tasks (client side) and the partial aggregation and optimal weight computation (coordinator side). This led to faster processes under certain scenarios.

As is common with other secure FL methods, a significant overhead arises from the implementation of the FHE encryption scheme, particularly affecting step 4 of the proposed algorithm – computation of global optimal parameters – where encrypted matrix-vector multiplication occurs. However, the incremental aggregation process over client-queued data remains computationally efficient, as the distributed computation of global matrices $\boldsymbol{U}$ and $\boldsymbol{S}$, along with the chained additions of encrypted vectors $[[\,\boldsymbol{m}_p\,]]$, are very efficient and lightweight by comparison. In scenarios like IoT environments with low computational power devices, it may be the only viable option for training clients on edge devices.

As future work, we highlight two lines of research: the first is experimental, regarding performance comparison between existing CKKS implementations, exploring current compatibility with ARM-based devices, and integrating libraries such as *Google Jax* to accelerate SVD computations — critical for data aggregation and training in the proposed method. The second is theoretical, focused on developing a more powerful model by combining this algorithm with other federated methods and designing a smart fine-tuning strategy of the Random Patches hyperparameters for any given dataset.

## ACKNOWLEDGMENT

Project PID2023-147404OB-I00 funded by MCIN/AEI/10.13039/501100011033/ with ERDF of the European Union. Also, the authors are supported by the Xunta de Galicia (Grant ED431C 2022/44) with the European Union ERDF funds and CITIC, as Research Center accredited by Galician University System, is funded by "Consellería de Cultura, Educación e Universidade from Xunta de Galicia", supported in an 80% through ERDF Operational Programme Galicia 2014-2020, and the remaining 20% by "Secretaría Xeral de Universidades" (Grant ED431G 2023/01).

## REFERENCES

[1] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council.

[2] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[3] Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

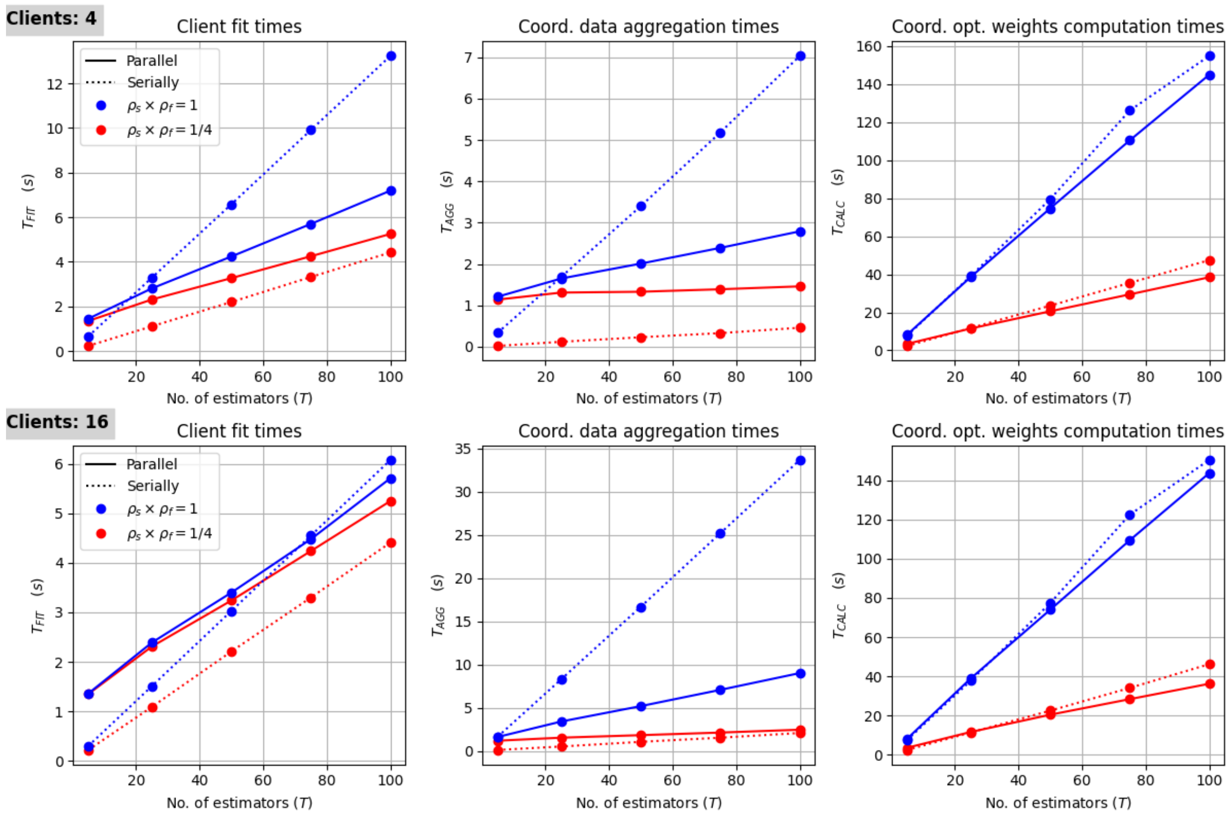

Fig. 2. Parallelization performance of FedHEONN-ensemble processes for different no. of estimators and relative patch size with $P = \{4, 16\}$ clients. Both client fitting and coordinator aggregation tasks benefit from multiprocessing at moderate number of estimators and high relative patch size, diverging from their serial counterparts. Solution computation does not reflect such differences.

[4] Gilles Louppe and Pierre Geurts. Ensembles on Random Patches. In *Machine Learning and Knowledge Discovery in Databases*, pages 346–361, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas, Beatriz Pérez-Sánchez, and Amparo Alonso-Betanzos. A new convex objective function for the supervised learning of single-layer neural networks. *Pattern Recognition*, 43(5):1984–1992, 2010.

[6] Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas, and Beatriz Pérez-Sánchez. Regularized One-Layer Neural Networks for Distributed and Incremental Environments. In *Advances in Computational Intelligence*, pages 343–355, Cham, 2021. Springer International Publishing.

[7] Oscar Fontenla-Romero, Bertha Guijarro-Berdiñas, Elena Hernández-Pereira, and Beatriz Pérez-Sánchez. FedHEONN: Federated and homomorphically encrypted learning method for one-layer neural networks. *Future Generation Computer Systems*, 149:200–211, 2023.

[8] Mark A. Iwen and Benjamin W. Ong. A Distributed and Incremental SVD Algorithm for Agglomerative Data Analysis on Large Networks. *SIAM Journal on Matrix Analysis and Applications*, 37(4):1699–1718, 2016.

[9] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 603–618, New York, NY, USA, 2017. Association for Computing Machinery.

[10] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[11] Shereen Mohamed Fawaz, Nahla Belal, Adel ElRefaey, and Mohamed Waleed Fakhr. A Comparative Study of Homomorphic Encryption Schemes Using Microsoft SEAL. *Journal of Physics: Conference Series*, 2128(1):012021, dec 2021.

[12] Leo Breiman. Pasting Small Votes for Classification in Large Databases and On-Line. *Machine Learning*, 36(1):85–103, Jul 1999.

[13] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[14] Fabio Mendoza-Palechor and Alexis Hoz-Manotas. Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico. *Data in Brief*, 25:104344, 2019.

[15] Ethem Alpaydin and Cenk Kaynak. Optical Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50P49.

[16] Murat Koklu and Ilker Ali Ozkan. Multiclass classification of dry beans using computer vision and machine learning techniques. *Computers and Electronics in Agriculture*, 174:105507, 2020.

[17] Byron Roe. MiniBooNE particle identification. UCI Machine Learning Repository, 2005. DOI: https://doi.org/10.24432/C5QC87.

[18] Rajen Bhatt and Abhinav Dhall. Skin Segmentation. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C5T30C.

[19] I Gede Susrama Mas Diayasa, Mohammad Idhom, Akhmad Fauzi, and Aviolla Terza Damaliana. Stacking Ensemble Methods to Predict Obesity Levels in Adults. In *2022 IEEE 8th Information Technology International Seminar (ITIS)*, pages 339–344, 2022.

[20] Mohammad Salauddin Khan, Tushar Deb Nath, Mohammad Murad Hossain, Arnab Mukherjee, Hafiz Bin Hasnath, Tahera Manhaz Meem, and Umama Khan. Comparison of multiclass classification techniques using dry bean dataset. *International Journal of Cognitive Computing in Engineering*, 4:6–20, 2023.

[21] Junhai Zhai, Ta Li, and Xizhao Wang. A cross-selection instance algorithm. *Journal of Intelligent & Fuzzy Systems*, 30(2):717–728, 2016.

[22] Boshra Pishgoo, Ahmad Akbari Azirani, and Bijan Raahemi. A hybrid distributed batch-stream processing approach for anomaly detection. *Information Sciences*, 543:309–327, 2021.