2024 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES) | 979-8-3315-4201-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICARES64249.2024.10767941

Federated Forests with Differential Privacy for Distributed Wearable Sensors

Manuele Favero
Dept. of Information Engineering
University of Padova, Italy
manuele.favero@studenti.unipd.it

Thomas Marchioro
Dept. of Information Engineering
University of Padova, Italy
thomas.marchioro@unipd.it

Chiara Schiavo
Dept. of Information Engineering
University of Padova, Italy
chiara.schiavo.2@studenti.unipd.it

***Abstract*—Training machine learning models on wearable sensor data is useful for applications like human activity recognition (HAR), but the sensitivity of such data often precludes centralized data collection. Federated learning offers a decentralized solution, enabling model training across distributed data sources. To further protect privacy, differential privacy (DP) can be integrated into the federated learning process. In this paper, we introduce a novel algorithm for training tree ensemble models (forests) within a federated learning framework under differential privacy constraints. Our approach allows each client to independently train a differentially private decision tree with randomized splits, which is sent to a central server. The server aggregates these trees into an ensemble that classifies data points via majority voting. We evaluate our DP federated forest algorithm on a HAR task, demonstrating that an effective privacy-utility tradeoff can be achieved with an appropriate selection of the privacy budget.**

***Index Terms*—federated learning, differential privacy, ensemble models, decision trees.**

I. Introduction

Wearable sensor data play a key role in many machine learning (ML) applications, particularly in human activity recognition (HAR). The success of these applications relies on collecting data from a large and diverse pool of individuals to ensure that the learned models generalize across different populations. However, the sensitive nature of wearable sensor data — ranging from raw measurements like accelerometer and gyroscope readings [1] to less fine-grained data such as step counts and calorie consumption [2] — raises significant privacy concerns. These concerns are further exacerbated by stringent data protection regulations like GDPR, making the traditional centralized data collection and model training paradigm increasingly untenable [3].

In response to these challenges, decentralized training solutions like federated learning have gained significant traction [4]. Federated learning enables models to be trained locally on edge devices (clients), with the resulting models aggregated by a central server [5]. While federated learning addresses the issue of data centralization, it is not inherently sufficient to guarantee privacy. Research has demonstrated that ML models can memorize and inadvertently leak personal information [6], with the potential to reconstruct original training data from the models themselves [7].

This work was supported by the Italian PRIN project 2022PNRR "DIGIT4CIRCLE," project code P2022788KK.

To mitigate such risks, differential privacy (DP) [8] has emerged as the de facto standard to prevent the memorization of individual data points during training [9]. DP introduces calibrated noise into the training process, thus ensuring that the presence or absence of any single data point does not significantly affect the model's output. The privacy-utility tradeoff in DP is controlled by a parameter known as the privacy budget, where a smaller privacy budget offers stronger privacy guarantees but at the cost of model utility. Importantly, it has been shown that even a relatively loose privacy budget can still provide substantial privacy protection in practice [6].

Although federated learning has been predominantly applied to train neural networks via federated averaging [4], recent works have explored its application to simpler models such as support vector machines [10] and Naive Bayes [11], [12]. These models typically yield a favorable privacy-utility tradeoff when differential privacy is applied. In this paper, we introduce a novel pipeline for training federated forest models under differential privacy. The term "federated forest" refers to an ensemble of decision trees across different clients, which can be built in a number of different ways [13], [14]. In our model, each client trains a differentially private decision tree, preventing the tree from memorizing individual data points during training. Then, the trees are shared and aggregated into an ensemble that makes its final decision via majority voting [15]. Our approach differs from existing DP algorithms for decision trees [16], [17] by incorporating random feature thresholding, an approach that has been shown to improve generalization in non-DP models such as Extra Trees [18].

After detailing our training algorithm and proving its differential privacy guarantees, we conduct an extensive evaluation using a HAR dataset. Our results demonstrate that using trees with limited depth (e.g., $d = 5$) allows for a favorable privacy-utility tradeoff. Furthermore, we show that even relatively loose values of the privacy parameter ϵ (e.g., $\epsilon = 10$) offer effective protection without significantly compromising performance. All the code used in our experiments is public: https://github.com/thomasmarchioro3/FederatedForestsWithDP.

II. Background and related work

A. Decision trees

Decision trees [16], [17] are ML models that can be applied to both classification and regression tasks; however, this paper

focuses specifically on classification. Decision trees classify data points by iteratively comparing different features x_f against specific thresholds $x_{f,\text{th}}$. Structurally, they follow a binary tree format, where each internal node represents a decision rule of the form $x_f \leq x_{f,\text{th}}$. This rule determines whether a data point is routed to the left or right branch of the node. The process continues until the data point reaches a leaf node, which is associated with a particular class. That class label becomes the final prediction for the data point.

During the training phase, the tree begins as a single root node and grows by iteratively splitting nodes based on a certain criterion, continuing this process until a predefined maximum depth is reached or no further meaningful splits can be made. Among the various criteria used to determine the best decision rule at each node, the most widely adopted is the Gini impurity index. Gini impurity measures the degree of class homogeneity within a node based on the distribution of the training data. If $p_1, \ldots, p_m$ the distribution of the m classes of the training data at a given node, the Gini index is computed as

$$I = 1 - \sum_{c=1}^{m} p_c^2. \tag{1}$$

Values $p_1, \ldots, p_m$ can be estimated by the class counts as $p_c = n_c / \sum_{c'=1} n_{c'}$. In standard decision trees, the possible decision rules at each node consist in comparing each feature against its median. In some ensemble models such as Extra Trees [18], the feature thresholds are chosen using a random data point from the training data, to reduce the model bias.

B. Federated forests

The term "federated forest" [13] is used for any ensemble of decision tree models created through the collaborative efforts of multiple clients in a decentralized system. In the context of this paper, training a federated forest involves each client independently training a decision tree with differential privacy applied to the process. Once trained, the clients share their trees with a central server, which stores the trees and utilizes them collectively to make predictions.

In scenarios where differential privacy is not employed, additional techniques, such as secure multi-party computation (SMPC), are often used to minimize privacy leakage to the central server [13], [14]. However, these methods do not provide formal theoretical guarantees of model privacy, unlike differential privacy.

C. Differential privacy

At its core, differential privacy is a statistical property that an algorithm can achieve through the introduction of randomization with respect to its input data. An algorithm $\mathcal{A}$ is said to satisfy ϵ-differential privacy if, for any two datasets differing by only a single entry, the output distribution of the algorithm changes by at most a factor of ϵ, i.e.,

$$\Pr[\mathcal{A}(\mathcal{D}) \in \mathcal{O}] \leq \epsilon \Pr[\mathcal{A}(\mathcal{D}') \in \mathcal{O}], \forall \mathcal{O} \subseteq \text{Range}(\mathcal{A}) \tag{2}$$

Typically, differential privacy is applied to datasets where each record corresponds to a different user, ensuring that the membership of any individual user in the dataset cannot be inferred. However, in the context of wearable sensor data, differential privacy can instead be used to prevent the memorization of individual records rather than to conceal the presence of a user altogether.

Standard values for ϵ vary depending on the application: for statistical queries (e.g., sum, average, median), ϵ is typically set to 1, while for machine learning models, ϵ values around 10 are common [19]. It is also important to note that even relatively large values of ϵ (e.g., $\epsilon = 100$) have been shown to effectively limit the memorization of data in certain models, such as large language models (LLMs) [6].

An important property of differential privacy is that the privacy parameter, ϵ, can be distributed across multiple randomized queries conducted on the same dataset $\mathcal{D}$. Specifically, if an algorithm needs to perform n queries, randomizing each query with ϵ/n-differential privacy ensures that the algorithm achieves overall ϵ-differential privacy. For this reason, ϵ is often referred to as the privacy budget of an algorithm.

In this paper, we focus on training decision trees under differential privacy, which involves running counting queries to estimate class counts at each node. Since the counts for different classes require to query distinct subsets of the training data, estimating them at a single node is considered a single query. However, the total privacy budget must be allocated across all possible features and nodes in the tree.

The Laplace mechanism is typically employed to enforce ϵ-differential privacy for a counting query. The mechanism simply consists in adding a random value (noise) to the true class count n_c. The random value is drawn from a Laplace distribution with mean 0 and scale $1/\epsilon$. Since the output of the mechanism may be a negative value, in this work we consider a clipped version of the Laplace mechanism that maps negative values to 0, i.e.,

$$L^+(n_c, \epsilon) = \max(n_c + \xi, 0),\ \xi \sim \text{Lap}\left(0, \frac{1}{\epsilon}\right), \tag{3}$$

which was also employed in other related works [20]. It is worth noting that the clipping is a simple post-processing operation and therefore does not alter the differential privacy guarantee enforced by the Laplace mechanism [21]. Another noteworthy remark is that the scale of Laplace noise is inversely proportional to the value of ϵ. Therefore, a small value of ϵ causes the noise to be larger, which in turn makes the estimated counts less accurate [15]. Furthermore, splitting the value of ϵ across multiple queries increases the scale of the noise that is applied to each individual query. In other words, running a large number of queries tends to degrade the performance of the overall algorithm.

III. Algorithm design

In this section, we describe the local training procedure that each client follows to build an ϵ-DP decision tree on their private dataset. Once the trees are trained locally, they are collected by a central server and aggregated into a federated

forest. This forest classifies data points by combining the decisions of individual trees through majority voting.

The training procedure is similar to the standard approach for decision trees, but with three key modifications: the tree's maximum depth is always fixed; the feature thresholds considered for splitting each node are randomly selected from predefined ranges $(x_{1,\min}, x_{1,\max}), \ldots, (x_{F,\min}, x_{F,\max})$; the class counts at each node are perturbed using the clipped Laplace mechanism described by eq. (3). Fixing the tree's depth is necessary for distributing the privacy budget across all queries made during training. This constraint must be determined before the algorithm is executed to ensure that the total number of queries does not exceed the allocated privacy budget. At each node, splits can be made based on F different features. If the maximum depth is set to d, the tree can have up to 2^d nodes, leading to a maximum of $2^d F$ queries. To achieve overall ϵ-differential privacy, each counting query is perturbed using the clipped Laplace mechanism with a privacy budget of $\epsilon' = \epsilon/(2^d F)$. Randomly selecting feature thresholds serves two purposes: it reduces bias in the ensemble model and avoids the need for additional queries to the training data, which would be necessary if thresholds were chosen based on data distribution (e.g., using the median). A complete description of the local training procedure is provided in Algorithms 1, 2, and 3. Algorithm 1 outlines the initial steps of the training process, where the privacy budget is allocated based on the selected maximum depth, and a differentially private estimate of the class distribution is performed at the root node.

Algorithm 1 Local decision tree training with DP

Require: Local training dataset $\mathcal{D}_i$, maximal depth d, privacy budget ϵ, feature threshold ranges $(x_{f,\min}, x_{f,\max})$ for $f = 1, \ldots, F$

1: $\epsilon' \leftarrow \epsilon/(2^d F)$
2: **for** $c = 1, \ldots, m$ **do**
3: $\quad n_c^{(0)} \leftarrow \textsc{Count}(i : y_i = c)$
4: $\quad \tilde{n}_c^{(0)} \leftarrow L^+(n_c, \epsilon')$
5: $e^{(0)} \leftarrow \textsc{ExpandNodeDP}(0)$ {root}
6: **return** root $e^{(0)}$

The algorithm then calls the recursive function ExpandNodeDP, detailed in Algorithm 2, at each new node $e^{(\ell)}$. This function takes several parameters as input, including the noisy class counts $\tilde{n}_1^{(\ell)}, \ldots, \tilde{n}_m^{(\ell)}$ at the current node, the subset of training data assigned to the current node $D_i^{(\ell)}$, the depth of the current node $d^{(\ell)}$, the maximum depth d, and the feature threshold ranges (defined by their boundaries $x_{f,\min}, x_{f,\max}$). As these parameters are either global or within the scope of the current node, the pseudocode denotes calls to ExpandNodeDP using only the node index ℓ as input, keeping the other parameters implicit for the sake of readability. The same holds for the FindBestSplitDP function.

The ExpandNodeDP function first estimates the Gini index $I^{(\ell)}$ from the noisy class counts. If the Gini index is 0 (meaning that the node is pure) or the maximum depth has been reached, the node becomes a leaf and the recursion stops. In this case, the predicted class is the one with the highest class count. Otherwise, the FindBestSplitDP function is called for the current node, and the recursion continues by invoking ExpandNodeDP on two newly created child nodes.

Algorithm 2 ExpandNodeDP

Require: Node index ℓ, noisy class counts at current node $\tilde{n}_1^{(\ell)}, \ldots, \tilde{n}_m^{(\ell)}$, local training dataset at current node $\mathcal{D}_i^{(\ell)}$, current depth $d^{(\ell)}$, maximal depth d, feature threshold ranges $(x_{f,\min}, x_{f,\max})$ for $f = 1, \ldots, F$

1: $I^{(\ell)} \leftarrow \textsc{Gini}(\tilde{n}_1^{(\ell)}, \ldots, \tilde{n}_m^{(\ell)})$ {Noisy counts $\tilde{n}_1^{\ell}, \ldots, \tilde{n}_2^{\ell}$ are estimated by Algorithm 1 for the root node and by Algorithm 3 for other nodes}
2: **if** $d^{(\ell)} \neq d \wedge I^{(\ell)} \neq 0$ {split only if max depth has not been reached and node is not pure} **then**
3: $\quad \ell_1 \leftarrow d^{(\ell)} + 2\ell$ {Left node index}
4: $\quad \ell_2 \leftarrow d^{(\ell)} + 2\ell + 1$ {Right node index}
5: $\quad f^{(\ell)}, x_{f,\text{th}}^{(\ell)}, \mathcal{D}_i^{(\ell_1)}, \mathcal{D}_i^{(\ell_2)}, \cdots \leftarrow \textsc{FindBestSplitDP}(\ell)$
6: $\quad e^{(\ell_1)} \leftarrow \textsc{ExpandNodeDP}(\ell_1)$
7: $\quad e^{(\ell_2)} \leftarrow \textsc{ExpandNodeDP}(\ell_2)$
8: **return** current node $e^{(\ell)}$

The FindBestSplitDP function (Algorithm 3) determines the best split for expanding the current node by considering a random threshold for each feature and evaluating the average Gini indices of the two child nodes ℓ_1, ℓ_2. The class counts for the left child are estimated by running a counting query, which is perturbed using the Laplace mechanism. For the right child, the class counts are estimated by subtracting the left child's counts from the total counts at the parent node. This approach helps to conserve the privacy budget by avoiding further distribution. The best split is the one that yields the lowest average Gini index between the two child nodes.

Theorem 1. *The execution of Algorithm 1 achieves ϵ-differential privacy.*

Proof. The training dataset D_i is queried once at the start of the process and then at most F times for every node that gets split (i.e., F times at each call of FindBestSplitDP). This implies that the number of counting queries cannot be greater than $2^d F$. Since each counting query is perturbed with the clipped Laplace mechanism to achieve ϵ'-DP with $\epsilon' = \epsilon/(2^d F)$, the overall procedure achieves ϵ-DP. □

IV. Dataset

To evaluate the performance of our federated forest algorithm, we utilized the HuGaDB dataset [22], which is widely used in the field of HAR. HuGaDB contains labeled sensor measurements recorded from 18 participants performing 12 different activities. The data was collected using six wearable inertial measurement units (accelerometers and gyroscopes) placed on the right and left thighs, shins, and feet. Additionally, two electromyogram (EMG) sensors were placed on the right and left quadriceps [23]. This dataset is suitable for training classification models that associate sensor measurements with corresponding activities [24], [25].

Algorithm 3 FINDBESTSPLITDP

Require: Node index ℓ, noisy class counts at current node $\tilde{n}_1^{(\ell)}, \ldots, \tilde{n}_m^{(\ell)}$, local training dataset at current node $\mathcal{D}_i^{(\ell)}$, feature threshold ranges $(x_{f,\min}, x_{f,\max})$ for $f = 1, \ldots, F$

$\bar{I}^{(\ell)} \leftarrow 1$ {Initialize average Gini index to its upper bound}
for $f = 1, \ldots, F$ **do**
  $x_{f,\text{th}} \leftarrow \zeta,\ \zeta \sim \mathcal{U}(x_{f,\min}, x_{f,\max})$
  for $c = 1, \ldots, m$ **do**
    $n_{c,f}^{(\ell_1)} \leftarrow \text{COUNT}(i : (x_i, y_i) \in \mathcal{D}_i^{(\ell)}, y_i = c \wedge x_{f,i} \leq x_{f,\text{th}})$ {Count classes on the left split for feature f}
    $\tilde{n}_{c,f}^{(\ell_1)} \leftarrow L^+(n_{c,f}^{\ell_1}, \epsilon')$
    $\tilde{n}_{c,f}^{(\ell_2)} \leftarrow \tilde{n}_c^{(\ell)} - \tilde{n}_{c,f}^{(\ell_1)}$
  $\bar{I}_f^{(\ell)} \leftarrow \dfrac{\text{GINI}(\tilde{n}_{1,f}^{(\ell_1)}, \ldots, \tilde{n}_{m,f}^{(\ell_1)}) + \text{GINI}(\tilde{n}_{1,f}^{(\ell_2)}, \ldots, \tilde{n}_{m,f}^{(\ell_2)})}{2}$
  if $\bar{I}_f^{(\ell)} < \bar{I}^{(\ell)}$ {Keep track of the minimum average Gini index} **then**
    $\bar{I}^{(\ell)} \leftarrow \bar{I}_f^{(\ell)}$
    for $c = 1, \ldots, m$ **do**
      $n_c^{(\ell_1)} \leftarrow n_{c,f}^{(\ell_1)}$
    $f^{(\ell)} \leftarrow f$
    $x_{f,\text{th}}^{(\ell)} \leftarrow x_{f,\text{th}}$
  $\mathcal{D}_i^{(\ell_1)} = \{(x_i, y_i) \in \mathcal{D}_i^{(\ell)} : x_{f,i} \leq x_{f,\text{th}}\}$
  $\mathcal{D}_i^{(\ell_2)} = \{(x_i, y_i) \in \mathcal{D}_i^{(\ell)} : x_{f,i} > x_{f,\text{th}}\}$
  return $f^{(\ell)}, x_{f,\text{th}}^{(\ell)}, \mathcal{D}_i^{(\ell_1)}, \mathcal{D}_i^{(\ell_2)}, \underbrace{\tilde{n}_1^{(\ell_1)}, \tilde{n}_1^{(\ell_2)}, \ldots, \tilde{n}_m^{(\ell_2)}}_{\text{Noisy class counts for } \ell_1, \ell_2}$

In our experiments, we isolated the data collected by each participant and treated it as the client's local data. In other words, each participant represents a client in our experimental setup. This approach mirrors a real-world scenario more closely compared to randomly splitting the dataset. To enable the aggregation of trees trained by different clients, we ensured that all clients shared the same subset of activities. Since some activities were performed only by a subset of participants, we limited our analysis to those that were performed by all clients. As a result, our dataset was reduced to 7 classes: 'Walking', 'Standing', 'Going up', 'Going down', 'Sitting', 'Standing up', and 'Sitting down'. The portion of the dataset containing these 7 classes was split into training and testing sets using a stratified random split to preserve the class distribution in both sets. The split was performed with a fixed PRNG seed (42) to ensure reproducibility.

The training data was divided into 18 local training datasets, each corresponding to one specific participant. The test data, however, was kept as a single large dataset comprising data from all participants. The test dataset is used solely to measure the impact of differential privacy on the model's performance, in terms of both accuracy and privacy.

V. EXPERIMENTAL EVALUATION

In our experiments, we formed federated random forests by aggregating differentially private decision trees trained according to Algorithm 1, using various values of the privacy budget: $\epsilon = 1$, 10, 100, and 1000, and maximum depths $d = 5$ and $d = 10$. Since differential privacy introduces noise, we repeated all experiments 10 times and reported error bars that reflect the average, worst, and best outcomes. Although this approach does not enable statistically significant comparisons (e.g., using p-values from statistical tests), it provides insight into performance variability for different values of ϵ and d.

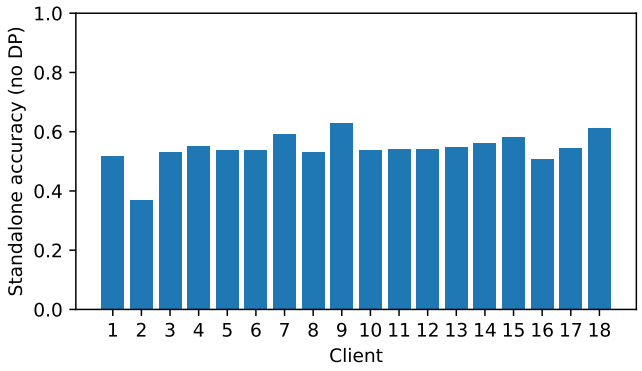

Fig. 1. Accuracy of local tree decision models trained by standalone clients with no DP.

To train differentially private decision trees on HuGaDB as described in section III, we had to select suitable feature threshold ranges $(x_{f,\min}, x_{f,\max})$. Accelerometer and gyroscope measurements take values as a (signed) 16-bit integer [22]. These measurements represent a quantization of the range $(-2g, 2g)$ with $g = 9.81\text{m/s}^2$ for the accelerometer, and $(-2000, 2000)\text{rad/s}^2$ for the gyroscope. EMG sensors, instead, use an unsigned 8-bit integer representation. In our experimental evaluation, we set threshold ranges to $(-5000, 5000)$ for both accelerometer and gyroscope measurements, and we $(0, 100)$ for the EMG measurements.

A. Utility assessment

The primary metric we evaluated to assess utility was accuracy, defined as the proportion of correct predictions on the test dataset, which measures the overall model performance. To assess the effectiveness of the federated forest, we compared its accuracy to a baseline in which clients trained non-DP models independently without sharing them. The accuracy values for each client on the test data are shown in Fig. 1. When training independently, most clients achieved around 55% accuracy, with a few reaching a top accuracy of 62-63%. Therefore, in our experiments, we consider accuracy values above 70% to be an improvement over standalone training.

As expected, the accuracy of the federated forest model increased with the privacy budget, as a higher privacy budget implies less noise in the counting queries and, consequently, looser privacy guarantees. Additionally, accuracy decreased as the maximum tree depth increased. This result is also unsurprising, as the number of counting queries increases exponentially with tree depth, leading to a proportional increase

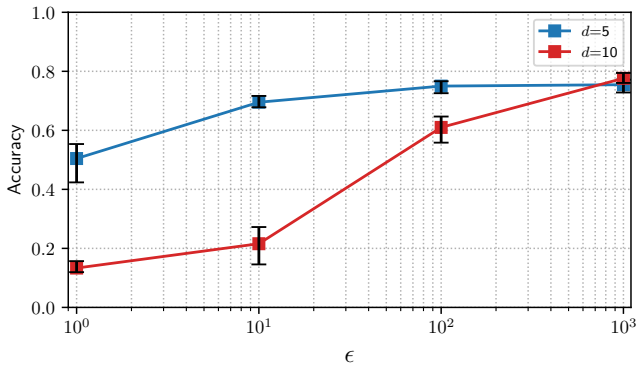

Fig. 2. Accuracy of federated forest models for $\epsilon = 1, 10, 100, 1000$. The line represents the average accuracy values across 10 trials, while the error bars show the minimum and the maximum.

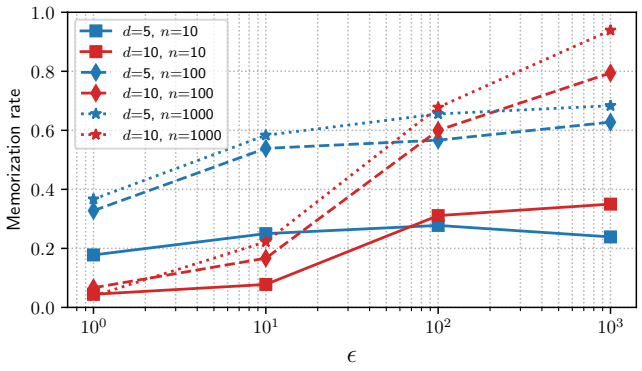

Fig. 3. Memorization rate of federated forest models for $\epsilon = 1, 10, 100, 1000$ tested using $n = 10, 100, 1000$ samples.

in the noise magnitude applied to each query. Clearly, a smaller number of precise counting queries results in a more effective model compared to a larger number of imprecise ones.

The accuracy curves for $d = 5$ and $d = 10$ are shown in Fig. 2. When the maximum depth is set to $d = 5$, a privacy budget of $\epsilon = 10$ already results in an improvement over standalone training, with accuracy around 70%. Moreover, when ϵ exceeds 100, accuracy reaches approximately 75%. However, for $d = 10$, accuracy does not surpass 70% until $\epsilon = 1000$, due to the number of counting queries being on the order of 10^4.

B. Privacy assessment

The above results suggest that a low tree depth is generally preferable when training differentially private federated forests. Nevertheless, it is important to note that for a large number of queries, the differential privacy bound becomes increasingly loose. For this reason, many works in differentially private machine learning employ additional metrics to evaluate the privacy level achieved by a mechanism. For instance, in the context of language models, a common method for assessing the effectiveness of differential privacy is to test the model's ability to memorize a "canary" phrase. However, the canary approach is applicable only to generative models, as their output is data. In the case of classification models, determining memorization is more challenging.

In this paper, we assess the memorization rate of individual tree models in a federated forest by comparing their accuracy on test data across different clients. If the tree trained by the i-th client for $i = 1, \ldots, N$ has higher accuracy on the test data provided by that same client compared to the other clients, we consider this an indicator that the model has memorized some information specific to the client but not relevant to the general task. We then compute the memorization rate as the fraction of clients in the forest whose model accuracy on their own test data is higher than that on the test data of all other clients. While the memorization rate is expressed as a percentage, it should not be interpreted as a success rate of extraction attacks, as is done with canaries in generative models. A possible interpretation could be the success rate of a model re-identification attack, i.e., the likelihood that an attacker could determine which client a tree belongs to. However, this does not necessarily imply that the attacker would be able to extract sensitive information about the client from the tree. This metric is used to provide a rough quantitative measure of the likelihood that a tree model memorizes client-specific information. Since the memorization rate may vary depending on the number of examples per user, we evaluate it using a fixed number of samples n per client, with $n = 10$, 100, and 1000. The results, averaged over 10 trials, are displayed in Fig. 3. One clear observation is that the memorization rate increases with the privacy budget, reaching values close to 1 when ϵ is very large. This is simply a consequence of the privacy-utility tradeoff established by differential privacy: larger values of ϵ allow for higher accuracy but reduce privacy guarantees.

Interestingly, for larger values of ϵ such as $\epsilon = 100$ and $\epsilon = 1000$, trees with a depth of $d = 5$ exhibit a lower memorization rate compared to those with $d = 10$. For $\epsilon = 10$, $d = 10$ provides better privacy but at the cost of significantly degraded accuracy. Overall, it appears that shallow trees offer a more advantageous privacy-utility tradeoff compared to deeper trees. This can be explained by the relationship between depth and overfitting: a deep tree is likely to memorize specific characteristics of the training data that do not generalize well to other samples.

VI. Discussion

a) Potential Improvements: While our experiments demonstrate that the federated forest model outperforms standalone decision trees in its current form, there is still room for enhancement. For instance, a more sophisticated selection of feature threshold ranges could reduce the likelihood of selecting uninformative thresholds during node splitting. Additionally, incorporating preliminary feature selection methods, such as mutual information [26] or ANOVA F-tests [27], could

help reduce the number of features and, consequently, the number of queries required during training.

b) Alternative Applications: Although we focused on sensor data and HAR as the primary case study for our federated forest algorithm, the training procedure is application-agnostic and can be applied to other machine learning tasks involving sensitive data. Future research could explore applications in areas like EEG and EMG-based classification tasks [28], where differentially private trees could be trained on data generated by different clients.

c) Decentralization: In section III, we introduced our forest model within the framework of federated learning, where a central server aggregates differentially private decision trees trained by clients. However, because privacy is enforced directly on the decision trees, this model is also well-suited for peer-to-peer applications, such as gossip learning [29].

d) Energy Efficiency: A significant advantage of our algorithm over more complex models, such as federated neural networks, is that it requires only a single exchange. This is crucial for energy efficiency, which is increasingly important in ML applications, particularly when edge devices are involved [30], as is the case with federated learning.

VII. Conclusions

We presented and evaluated a method for constructing differentially private federated forests by aggregating decision trees trained locally by individual clients. Our experiments focused on HAR tasks and demonstrated that limiting tree depth can significantly improve the privacy-utility tradeoff, with low-depth trees offering better protection against client-specific memorization while maintaining higher accuracy. While our results highlight the potential of differentially private federated forests, several challenges and opportunities for future research remain. In particular, further investigation into more sophisticated methods for feature selection and threshold determination under differential privacy could enhance model performance [31].

References

[1] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Mobile sensor data anonymization," in *Proc. Int. Conf. Intern. Things Design Implem.*, pp. 49–58, 2019.

[2] A. Kazlouski, T. Marchioro, and E. Markatos, "What your Fitbit says about you: De-anonymizing users in lifelogging datasets," in *Proc. Int. Conf. Sec. Crypt. (SECRYPT)*, vol. 1, pp. 341–348, 2022.

[3] L. Giaretta, T. Marchioro, E. Markatos, and Š. Girdzijauskas, "Towards a decentralized infrastructure for data marketplaces: narrowing the gap between academia and industry," in *Proc. Int. Wkshp Data Econ.*, pp. 49–56, 2022.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artif. Intell. Stat.*, pp. 1273–1282, PMLR, 2017.

[5] A. Buratto, A. Mora, A. Bujari, and L. Badia, "Game theoretic analysis of AoI efficiency for participatory and federated data ecosystems," in *Proc. IEEE Int. Conf. Commun. (ICC) Workshops*, pp. 1301–1306, 2023.

[6] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *USENIX Secur. Symp.*, pp. 267–284, 2019.

[7] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, *et al.*, "Extracting training data from large language models," in *USENIX Secur. Symp.*, pp. 2633–2650, 2021.

[8] C. Dwork, "Differential privacy," in *Int. Colloq. Automata Lang. Progr.*, pp. 1–12, Springer, 2006.

[9] A. El Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE Access*, vol. 10, pp. 22359–22380, 2022.

[10] D. G. Nair, C. Aswartha Narayana, K. Jaideep Reddy, and J. J. Nair, "Exploring svm for federated machine learning applications," in *Proc. Int. Conf. Adv. Distrib. Comput. Machine Learn. (ICADCML)*, pp. 295–305, Springer, 2022.

[11] T. Marchioro, L. Giaretta, E. Markatos, and S. Girdzijauskas, "Federated naive Bayes under differential privacy," in *Proc. Int. Conf. Sec. Crypt. (SECRYPT)*, pp. 170–180, 2022.

[12] M. M. Rahman and D. M. Farid, "Exploring federated learning with naïve Bayes using AVC information," in *Proc. IEEE Int. Conf. Comput. Commun. Netw. Tech. (ICCCNT)*, pp. 1–6, 2023.

[13] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, "Federated forest," *IEEE Trans. Big Data*, vol. 8, no. 3, pp. 843–854, 2020.

[14] L. A. C. de Souza, G. A. F. Rebello, G. F. Camilo, L. C. Guimarães, and O. C. M. Duarte, "Dfedforest: Decentralized federated forest," in *Proc. IEEE Int. Conf. Blockchain*, pp. 90–97, 2020.

[15] A. Buratto, B. Yivli, and L. Badia, "Machine learning misclassification within status update optimization," in *Proc. IEEE Int. Conf. Commun. Netw. Satell. (COMNETSAT)*, pp. 640–645, 2023.

[16] S. Fletcher and M. Z. Islam, "Decision tree classification with differential privacy: A survey," *ACM Comput. Surv. (CSUR)*, vol. 52, no. 4, pp. 1–33, 2019.

[17] R. Hazra, M. Banerjee, and L. Badia, "Machine learning for breast cancer classification with ANN and decision tree," in *Proc. IEEE Ann. Inf. Tech. Elec. Mobile Commun. Conf. (IEMCON)*, pp. 0522–0527, IEEE, 2020.

[18] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, 2006.

[19] N. Ponomareva, H. Hazimeh, A. Kurakin, Z. Xu, C. Denison, H. B. McMahan, S. Vassilvitskii, S. Chien, and A. G. Thakurta, "How to DP-fy ML: A practical guide to machine learning with differential privacy," *J. Artif. Intell. Res.*, vol. 77, pp. 1113–1201, 2023.

[20] M. Lopuhaä-Zwakenberg, M. Alishahi, J. Kivits, J. Klarenbeek, G.-J. van der Velde, and N. Zannone, "Comparing classifiers' performance under differential privacy.," in *Proc. Int. Conf. Sec. Crypt. (SECRYPT)*, pp. 50–61, 2021.

[21] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comp. Sc.*, vol. 9, no. 3–4, pp. 211–407, 2014.

[22] R. Chereshnev and A. Kertész-Farkas, "Hugadb: Human gait database for activity recognition from wearable inertial sensor networks," in *Proc. Int. Conf. Anal. Imag. Soc. Netw. Texts (AIST)*, pp. 131–141, Springer, 2018.

[23] G. Cisotto, A. V. Guglielmi, L. Badia, and A. Zanella, "Joint compression of EEG and EMG signals for wireless biometrics," in *Proc. IEEE Global Commun. Conf. (Globecom)*, 2018.

[24] A. Kazlouski, T. Marchioro, H. Manifavas, and E. Markatos, "Do partner apps offer the same level of privacy protection? the case of wearable applications," in *Proc. IEEE PerCom Workshops*, pp. 648–653, 2021.

[25] A. Buratto, H. Tuwei, and L. Badia, "Optimizing sensor data transmission in collaborative multi-sensor environments," in *Proc. IEEE Int. Conf. Commun. Netw. Satell. (COMNETSAT)*, pp. 635–639, 2023.

[26] B. C. Ross, "Mutual information between discrete and continuous data sets," *PloS One*, vol. 9, no. 2, p. e87357, 2014.

[27] L. St, S. Wold, *et al.*, "Analysis of variance (anova)," *Chemom. Intell. Lab. Syst.*, vol. 6, no. 4, pp. 259–272, 1989.

[28] G. Cisotto, A. V. Guglielmi, L. Badia, and A. Zanella, "Classification of grasping tasks based on EEG-EMG coherence," in *Proc. IEEE Int. Conf. e-Health Netw. Appl. Serv. (Healthcom)*, 2018.

[29] L. Giaretta and Š. Girdzijauskas, "Gossip learning: Off the beaten path," in *Proc. IEEE Int. Conf. Big Data*, pp. 1117–1124, 2019.

[30] N. Michelusi, K. Stamatiou, L. Badia, and M. Zorzi, "Operation policies for energy harvesting devices with imperfect state-of-charge knowledge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 5782–5787, 2012.

[31] G. Ioannou, T. Marchioro, C. Nicolaides, G. Pallis, and E. Markatos, "Evaluating the utility of human mobility data under local differential privacy," in *Proc. IEEE Int. Conf. Mob. Data Manag. (MDM)*, pp. 67–76, 2024.