Scuola universitaria professionale della Svizzera italiana

Internal membership inference attack on private federated models

Author

Sander Peeters

Supervisors

Davide Andreoletti, Fatima Ezzeddine

Advisor

Michela Papandrea

Course

Master of Science in Engineering with specialty in Data Science

Module

Thesis

Year

2023/2024

Date

September 25, 2024

STUDENTSUPSI

Abstract

In this paper, a novel way of simulating a membership inference attack in a federated machine learning setup is proposed. The taken approach is to generate a shadow dataset using a diverse counterfactual explanation algorithm. After this data is generated, the adversary launches the attack from a malicious client and tries to obtain raw data from the other clients through a victim model, which in the experiments has various levels of differential privacy applied. The results indicate that differential privacy significantly impacts the performance of an attack and that data generated from a diverse counterfactual explanation algorithm can be used in an internal membership inference attack, but also that this method has certain limitations. Besides this, a performance analysis is done between centralized and federated models and the benefits to security and privacy of a federated setup are discussed.

Keywords: Federated learning, Membership inference attack, Counterfactuals, Differential privacy

Contents

1 Introduction

Security and privacy in machine learning models is becoming an increasingly important topic, especially now with more legislation being created around the topic in the General Data Protection Regulation (GDPR) of 2018 [1] and more recently, the EU AI act of December 9th, 2023 [2]. Even in developing countries such as Brazil, Lei Geral de Proteção de Dados (LGPD), a GDPR variant with extra-territorial provisions was introduced in 2020 [3]. These legislations were created to prevent unethical ways of dealing with sensitive data. The possible fines for large corporations are now in percentages of their turnover, instead of a fixed number. A result of this was Meta's 1.2 billion Dollar fine in 2023 [4], resulting in large corporations being more careful with compliance to these rules.

There are already various ways to increase the privacy of users of whom data is being collected for use in a machine learning model, one of them being federated learning. This setup consists of a server and a number of clients, who are the data owners. The data does not leave the clients, instead, each of them performs model training on their own dataset. Only the model parameters resulting from these local training rounds are shared with the server, where they are aggregated.

Federated setups are being adopted for a number of reasons; to start, it is a safer way to store and make computations on sensitive data. Since data is stored in a distributed way, if one storage device is breached not all data is leaked, possibly only a subset and a trained model. Sensitive data does not leave the data owner, this is good for their privacy preservation while at the same time supporting collaborative learning across multiple parties. Also, a federated setup is advantageous in situations with low bandwidth, since only the model parameters are shared and not the raw data which results in reduced overhead. In sectors with strict data protection regulations, federated learning provides a way to comply with privacy requirements while still benefiting from machine learning advancements. The distributed nature of federated learning aligns well with data protection principles. Multiple large companies that heavily rely on sensitive data such as healthcare, insurance and banking already adopted this strategy. An example is Johnson and Johson with their HONEUR project [16].

Even though federated learning is beneficial in many privacy and security aspects, it is not

completely privacy-preserving and introduces also some new risks [15]. Since the setup is distributed, it creates more possible access points for an adversary. When the security to a client device is compromised, an attacker can use this client to gain access to the model and exploit its information leakage in the model updates. By doing this, an attacker can infer whether specific data points are part of the original training dataset. This is called an Internal Membership Inference Attack (IMIA).

To counter this attack, one must minimize the contribution of individual records to the model and provide privacy guarantees. This can be achieved by applying Differential Privacy (DP) to the model. DP introduces structured noise to the model parameters, forcing the model to learn the patterns and not specific data points. DP can be applied to local training on the clients or to the aggregation process on the server. The focus of this research is to examine the effectiveness of differential privacy in preventing successful Membership Inference Attack (MIA) within a federated learning setup.

This research undertakes the simulation of an IMIA under varying DP configurations. In the initial phase of an IMIA, the adversary tries to construct a dataset analogous to the original training data by inferring it from the victim model, this data is referred to as a shadow dataset. The generation of such a dataset is achieved through the implementation of Diverse Counterfactual Explanations (DiCE). DiCE, as the name suggests, is an algorithm leveraging model explainability to generate counterfactual explanations. Specifically, explanations are derived using explainable AI techniques, which aim to elucidate the decision-making process of a model, and may unintentionally furnish attackers with valuable insights to bolster their attacks. This introduces a new challenge to existing security measures, as explanations intended for transparency could inadvertently facilitate adversarial efforts to exploit vulnerabilities in the model.

This is due to their proximity to the decision boundary and their representativeness of the training data. While counterfactuals serve as valuable interpretative tools, they also present opportunities for bolstering attacks by furnishing additional insights and knowledge to potentially exploit vulnerabilities in the targeted model.

Upon the acquisition of the counterfactual shadow dataset, a series of shadow models is trained, utilizing the same learning algorithm as the target model. These shadow models are designed to replicate the behavioral patterns of the victim model. Subsequently, using the predictions from the shadow models, an attack model is trained with the purpose of distinguishing between instances that were part of the shadow dataset and instances that were not based on the information learned from the shadow model. Following the training of the attack model, it is then applied to the original predictions from the victim model.

The contribution of this thesis can be summarized as follows:

- The observation that federated learning models require more time than centralized models to train a similar model, due to the fact that the server needs to wait for the clients to submit their updates, and then to aggregate them.
- The use of DP in both centralized and federated setups influences model performance, with the extent of impact contingent upon the level of privacy guarantee imposed. Specifically, the federated model experiences a more pronounced reduction in performance when subjected to DP compared to a centralized setup.
- the substantial impact of DP on the performance of IMIA's, rendering the attack impractical to the point of useless depending on the amount of DP applied.
- The possibility of using counterfactual explanations as shadow dataset for the use in an IMIA in a federated setup

The study is based on the following research questions:

- What is the impact on performance when switching from a centralized to a federated setup?
- What is the impact of applying DP to the performance of a federated and centralized setup?
- How good is DP in preventing a successful IMIA using counterfactual explanations?

The subsequent text outlines the structure of the remaining content in this work: A literature review of related work can be found in Section 3. Chapter 3 explains the theoretical concepts discussed in this research and their background. The problem formulation can be found in Chapter 4. Chapter 5 describes the methodology of how the concepts discussed in Chapter 3 are applied. In Chapter 6, the results of these experiments are shown and discussed. Conclusions are formed from the acquired results, and the study's limits and future work are discussed in Chapter 7.

2 | Related work

Federated learning has been proposed in [5], and applied in various contexts such as healthcare [6], finance [7], and telecommunications [8] to protect the privacy of data owners, such as personal health records and variables used for credit risk assessment such as income and current debts. Among the different federated learning approaches, horizontal federated learning has been applied in contexts where data owners possess the same features, related to different data points, such as in networking [8] and healthcare [6]. Vertical split learning has been applied in contexts where data owners possess different features related to the same data points, such as in [9] and [10]. On the topic of networks, horizontal split learning has been applied in [11], focusing on 5G networks. In this work, a horizontal federated learning approach is used aiming to classify good or bad connections. Despite being considered a privacy-preserving method, federated learning is also subject to attacks. Examples of works considering attacks on federated learning include [12], which investigates data poisoning attacks, [13], which explores model inversion attacks, and [14] which examines membership inference attacks (MIA).

In the following paper [23], the methodology used by the researchers for conducting a MIA focuses on a scenario where the attacker needs to classify if a given data record is part of the original training dataset when having black-box access to the trained victim model in a centralized setup. To perform membership inference, an attack model is trained to recognize differences in the victim model predictions based on its inputs. To train an attack model, the predictions of multiple shadow models were used, these models are in turn trained on a synthetic shadow dataset. The researchers investigated 3 ways of obtaining such a dataset. Using; 1) Noisy real data; 2) Statistics-based synthesis and 3) Model-based synthesis. The first 2 approaches require prior knowledge about the original training data while the 3rd method does not. In the first approach, the researchers assume a scenario where an attacker has access to data similar to the victim model their training data. This was simulated by inversing randomly selected binary features of the original training data. In the second approach, an adversary has statistical knowledge about the population from which the victim model their training data was taken. It was reported that attacks using this approach were very effective. The final and most complex approach uses the victim

model itself. This is done by using an optimization algorithm to search the space of certain features that are classified by the model with high confidence. And then, sampling synthetic data from their distribution.

The described 3rd approach used by the researchers is similar but simpler to the one used in this paper, the similarities are that also an optimization algorithm is applied to the victim model, but in this research it is used to create counterfactuals of the opposite class using a DiCE algorithm, posing a 'what-if?' question. The key aspect of DiCE is its ability to generate data covering a wide range of possible scenarios while adhering to user-defined constraints or preferences as described in the paper [32]. The diversity ensures that the generated counterfactual instances provide meaningful insights into the model its decision-making process and enable users to better understand the factors influencing the model its predictions. In the DiCE method, upon receiving a query vector along with its outcome class, diverse counterfactual explanations are computed by identifying candidate feature vectors closely resembling the query vector but having an opposite prediction. This differentiates from the third approach, where researchers do not make use of a query vector or opposite outcome classes. Instead, the researchers commence by initializing a random vector where each feature's value is sampled uniformly at random from the complete range of possible values for that feature and the outcome class is a fixed variable. The reason why the researchers have to use a random vector is because their attack is applied to a centralized setup, while in this paper the attack is applied to a federated setup in which one of the clients is compromised. As a result, the adversary has access to the local data from the client and thus can use a specific vector query. After obtaining the shadow dataset by using DiCE, the methodology for the internal MIA used in this paper is the same as described in [23] with the exception that the resulting attack model is used to determine the local data on each participating client, instead of data stored on a centralized server.

There are various ways to defend a model from an MIA attack, as described by [15]. The first strategy involves leveraging fully homomorphic encryption, wherein the model receives encrypted parameters as input, without necessitating decryption beforehand. A second tactic involves creating a more trusted environment or platform, this is especially relevant in a federated setup where the participating clients can be many and often not located in controlled environments. To prevent an adversary from gaining access to the setup, and executing an internal MIA, the client-side security cannot be overlooked. Finally, the most commonly used approach is DP, where structured noise is added to the model parameters.

This technique is exemplified in the following paper [19], wherein DP is applied to a federated setting. More specifically, in their paper, the researchers add DP during the training process on the clients, before aggregating the client their model parameters into the central server. Next, they investigate the following points: 1) the trade-off between model performance and privacy; 2) whether adding more clients to the federated learning setup improves the

performance ;3) whether using a selective process to determine which clients can participate in the aggregation per global training rounds affects the performance. Their methodology is similar to the one used in this paper, where DP is also added on the clients itself and not in the aggregation process, but their approach differs in the use of a selective process aggregation process, where not all clients participate in each aggregation round. Such a selective process was not applied in this paper.

3 Theoretical Background

In order to grasp the full extent of this paper, a comprehensive understanding of all discussed concepts is essential. The subsequent section will delve into the foundational principles of neural network models, followed by a section on their application within a federated setup. Subsequently, attention will be directed towards a prevalent vulnerability inherent in such setups, known as an IMIA. The section hereafter will provide a detailed elucidation of DiCE's, considering it is part of the novel contribution of this paper. Finally, a discussion on a preventive measure aimed at preventing such attacks, namely DP, will be presented.

3.1. Deep learning

Deep learning is a subfield of machine learning that utilizes deep neural networks to perform tasks such as regression and classification. These networks, characterized by their layered structure, enable intricate computations and pattern identification in data.

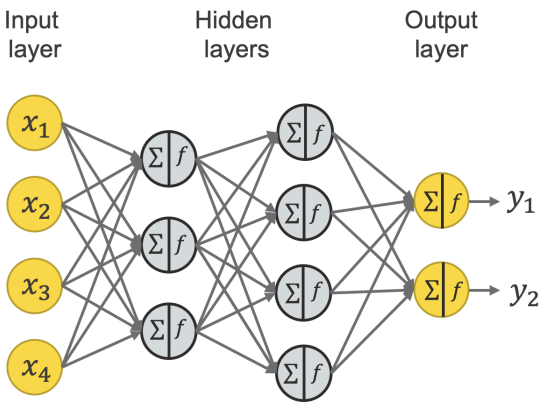


Figure 3.1: Structure of a Neural Network (source: [26])

At its foundation, a neural network comprises an input layer, several hidden layers, and an output layer. The input layer's nodes represent individual data features, as depicted in Figure 3.1. In the output layer, each node correlates to a target class or value for prediction.

For instance, in binary classifiers, there are typically two nodes, each symbolizing a distinct class.

The computations within neural networks occur primarily in the hidden layers. These layers process the incoming data by applying a set of model-specific weights and biases. The process of fine-tuning these parameters is pivotal for the accuracy of the model and is achieved through a method known as back-propagation [30]. Back-propagation entails the adjustment of the network's weights in response to the error margin between the predicted outputs and the actual ground truth.

Activation functions are integral to this process. Their role is to introduce non-linearity into the network, allowing it to capture complex patterns and relationships in the data, which would be impossible with linear operations alone. A commonly used activation function in binary classification is the Sigmoid function, whichis depicted below in Figure 3.2. It effectively compresses the output to a range between 0 and 1, making the output values interpretable as probabilities of belonging to one class or the other.

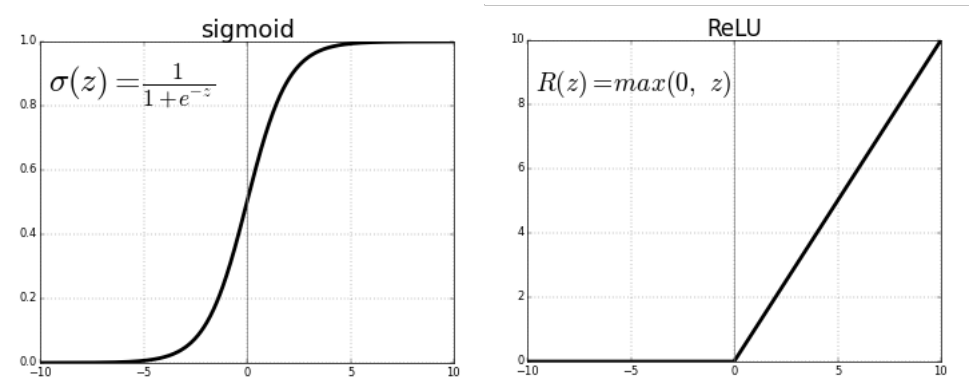


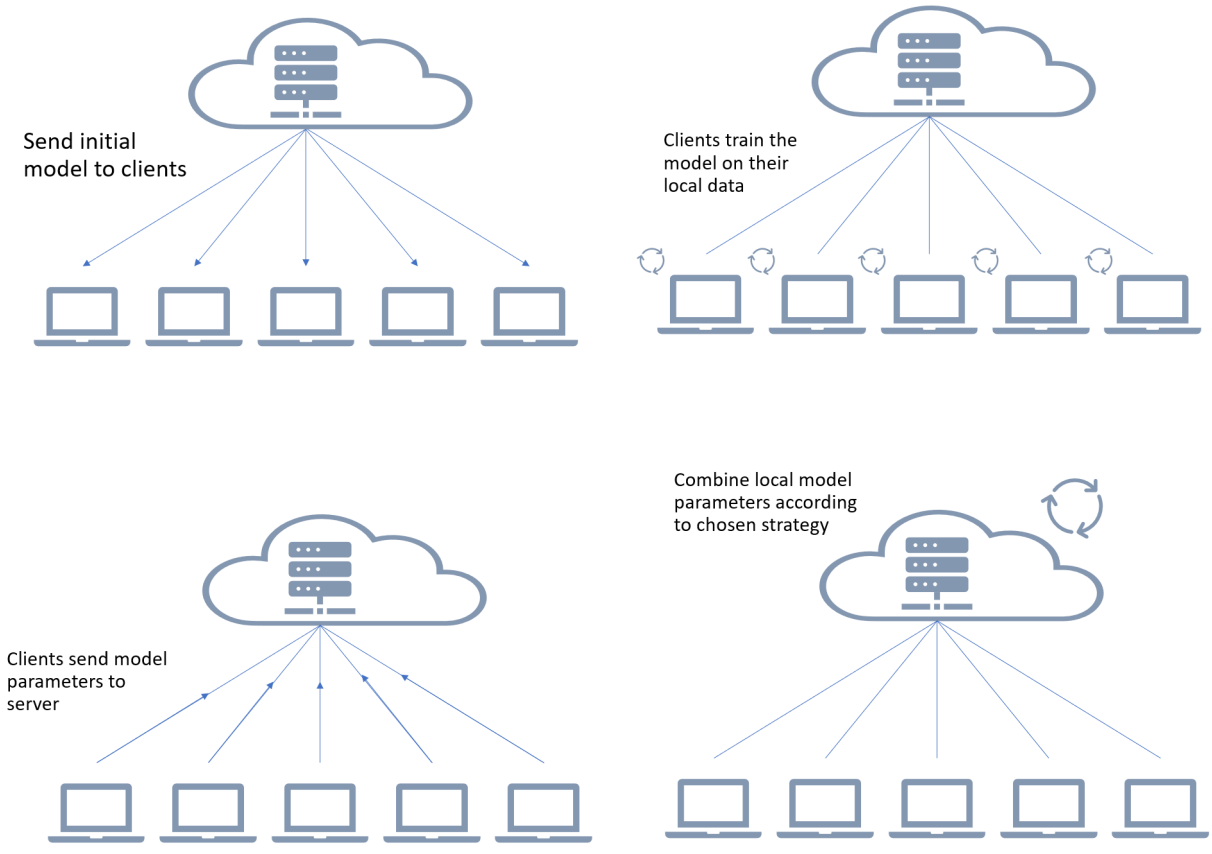
Figure 3.2: Sigmoid and ReLU Activation Functions (source: [28])

Another prevalent activation function is the Rectified Linear Unit (ReLU), frequently utilized in the hidden layers of deep neural networks. ReLU, visualized in 3.2, is favored in deep learning models because it helps overcome the vanishing gradient problem, a significant challenge in deep networks. This problem arises when the gradients used during back-propagation for updating weights become progressively smaller as they are propagated backwards through the network layers, hindering the learning process. ReLU effectively prevents this by maintaining a constant gradient for positive inputs, ensuring that the gradient remains significant for effective learning. In contrast, functions like Sigmoid or Hyperbolic Tangent tend to saturate with large positive inputs, leading to minuscule gradients [31].

By employing the described methods, deep learning neural networks are capable of efficiently learning models from complex datasets, making them suitable for a wide range of applications.

3.2. Federated learning

Traditionally, machine learning models (such as the deep neural networks described in the previous Section 3.1) are trained in a centralized manner, meaning all their training data exists in 1 centralized location and the model is also trained in 1 location. Federated learning marks a significant departure from traditional centralized machine learning methods. In this approach, the model training is decentralized, occurring across multiple devices, referred to as clients. These clients independently compute model parameters, which are subsequently transmitted to and aggregated on a central server. A representation of a federated learning architecture is depicted in 3.3.



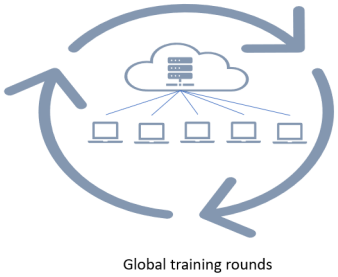


Figure 3.3: Federated Learning Process

One of the main advantages of a federated setup lies in its enhanced privacy features, particularly relevant when data is sourced from multiple personal devices. In this model, raw data remains exclusively on the user's device, never leaving its origin, making federated learning a privacy-preserving approach. For instance, in the event of a breach on one device, the compromised information is limited to that device only. Similarly, if the central server is hacked, no raw data can be extracted.

Federated learning can be implemented in two primary forms: horizontal and vertical splits. In horizontal federated learning, all clients possess the same features but different records. Conversely, in vertical federated learning, each client holds unique features, as would be the case in weather prediction scenarios where each device has a distinct sensor [33].

Training in federated learning comprises two phases: local and global. Local training describes the process where clients use their data to train the model. Global training, on the other hand, involves the aggregation of parameters on the central server. As illustrated in Figure 3.3, a typical global training round in federated learning begins with distributing the initial model from the central server to the clients. These clients then perform local training. Upon completion, they transmit their model parameters back to the server, where they are aggregated using various strategies. Popular strategies include averaging and median, but more advanced methods like adaptive federated optimization (FedAdam) are also available [22]. It is worth noting that including all clients in every aggregation round may not always be optimal for model performance, a topic explored in depth in recent studies [19].

Despite being a privacy-preserving approach, some privacy challenges persist in a federated scenario. One notable concern is the potential of reverse engineering model parameters to reconstruct raw data through a model inversion attack, or the potential of an attacker to infer if records are part of the original training data by exploiting models that generalize badly on new data, such a procedure is a membership inference attack and is described in the next section.

3.3. Membership Inference Attack

Membership inference attacks (MIA) are designed to determine if a specific record was included in a model's initial training set. These attacks exploit information leakage present in the model its outputs to infer whether a specific data point was included in the original training data. The attacker has access to the target model's parameters, its predictive outputs and in the case of a federated setup, also has access to the local data of the compromised client [20]. In that case, the attack is referred to as an internal membership inference attack.

The process of conducting a MIA is depicted in Figure 3.4. The first step involves creating a shadow dataset, essential for training shadow models. There exist various methods to obtain such a dataset, some of which require prior knowledge about the data while others use just the victim model and an optimization algorithm. In an internal membership inference attack, at least one client is compromised and thus the attacker has access to its local data. This local data can be used in combination with the optimization algorithm to create a shadow dataset. After obtaining this, it is used to train multiple shadow models that mimic the behavior of the victim model. The shadow models are used to infer whether specific instances were part of the training dataset used to train the target model

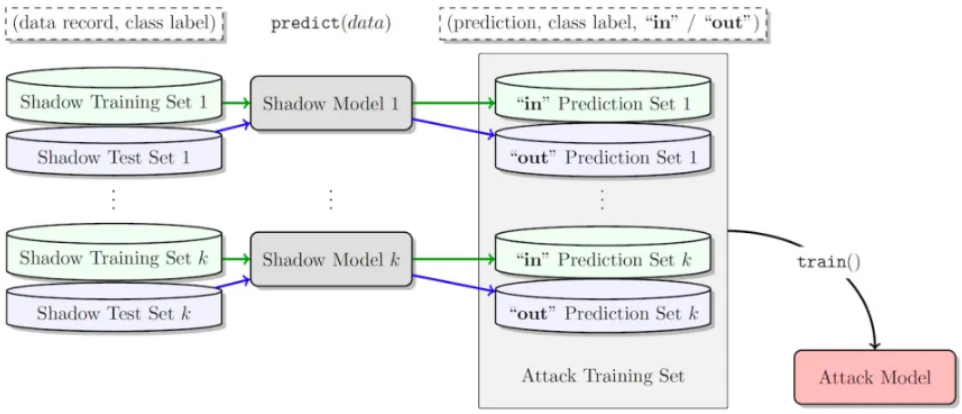


Figure 3.4: Process of a Membership Inference Attack (source: [20])

On the predictions of the shadow models, an attack model is trained to recognize differences in the victim model output based on its input. The trained attack model in turn can be applied to the victim model to classify whether each instance was likely to be part of the original training dataset.

Models that are poorly generalized, especially those that are overfitted, are more susceptible to MIAs. This vulnerability is particularly pronounced in models trained on tabular data. Conversely, models trained on highly-dimensional data (e.g., high-resolution images) are less vulnerable due to the increased complexity and time required to generate an effective

shadow dataset.

As previously mentioned, in a federated learning environment, an MIA can take the form of an internal attack, where a malicious client uses the aggregated model from the server to deduce information about data held by other clients. This scenario, is known as an internal membership inference attack. A practical example of such an attack could be in a network of hospital branches, each being a client in a federated learning system. Suppose these branches are reluctant to share patient data because of privacy concerns. In this case, an attacker infiltrating one branch could potentially use the federated model to infer data from the other branches, compromising patient privacy.

3.4. Diverse Counterfactual Explanations

Diverse counterfactual explanations (DiCE) refer to creating explanations that represent alternative scenarios to the observed data while maintaining diversity among them. This data can offer explanations about a black-box model. Asking the model, what would the input data look like if the outcome was a different class, for example, the opposite. Counterfactuals indicate the minimal changes required in input features to achieve a specific desired output from a model as described in [32].

How this algorithm works in detail can be seen in the pseudocode from Figure 3.5 below:

```
# Given
X = feature vector
f = model
C = desired class
q = amount of counterfactuals desired
#---------
function DiCE (X,f,C,q):
   CF = empty set
   counter = 0

   while counter < q:
      for each i in X:
         feature_range =  find the potential range of a feature i using an optimization algorithm.
         X' = Perturbed instance using feature i and the defined feature_range.

      for each perturbed instance x' obtained:
         difference = X' - X
         difference = adjust the difference according to C
         cf_x = difference + X

#Verify counterfactual
         prediction = f.predict(cf_x)
         if prediction == C:
            CF.add(cf_x)
      counter = counter + 1
   return CF
```

Figure 3.5: Pseudo code for diverse counterfactual explanation algorithm

In the first step, an empty set CF and an iterative loop are created which keeps running until the desired amount of instances is reached. Within this loop, there are 2 unnested loops. The first loops over the available features and by using an optimization algorithm, often gradient descent, the range of each feature is defined. in some cases, when domain knowledge is available then this can also be incorporated into the feature range at this point. Next, a vector X' is created including for each feature a perturbed value within the feature range.

In the second loop, the difference is calculated between the given feature vector X and the explanation vector X'. This difference is then adjusted to correspond with the correct desired class C. The adjusted difference is then added to the feature vector X. Before adding the explanation X' to the empty set CF, the counterfactual should be validated using the model to predict its class. If the predicted class equals the desired class, the explanation feature vector is added to the set.

As can be seen in the pseudocode, the algorithm loops over the features available in the given vector X. As a result, the running time is greatly influenced by the number of features in the data. Additinoally, the chosen optimization algorithm used to define the feature space also plays a factor.

As a result of this, applying the algorithm to high-dimensional data types such as images can be computationally demanding and the outcome of low quality. Counterfactual algorithms exhibit the flexibility to function both with and without the presence of specific query instances. In a slightly different algorithm where no feature vector is provided, a random vector is initialized where each feature's value is sampled uniformly at random from the complete range of possible values for that feature.

One notable application of counterfactuals is their role in the creation of shadow datasets. To illustrate this concept, suppose we have a machine learning model designed to evaluate mortgage applications. A person applies for a mortgage but receives a rejection due to their current financial situation, like the example in Table 3.1. By applying a counterfactual algorithm to that model, we can precisely identify the changes necessary in the individual's financial circumstances—such as household income and equity—to alter the decision from rejection to approval for the mortgage, as can be seen in Table 3.2. By having such additional information, it is possible to exploit counterfactuals to generate realistic shadow datasets for a wide range of applications, including the training of MIA models.

Query vector			
Household income	Equity	Desired amount	Mortgage approved
240.000	200.000	1.500.000	No

Table 3.1: Example of a feature vector with its outcome.

Counterfactual			
Household income	Equity	Desired amount	Mortgage approved
280.000	400.000	1.500.000	Yes

Table 3.2: Example of a counterfactual.

3.5. Differential privacy

Differential privacy (DP) is a technique applied to machine learning models to obscure specific data points while still enabling the model to discern general patterns. This approach significantly bolsters the model's resistance to overfitting and enhances its generalization capabilities [18]. A key aspect of DP is its contribution to both privacy and security. For instance, in the context of a Membership Inference Attack, the attack's success often hinges on the model's failure to generalize effectively. DP, by introducing structured noise into the data, mitigates this risk.

The essence of DP is to add noise to the data in a controlled manner. This concept is encapsulated in Cynthia Dwork's differential privacy formula (see Eq. 3.1). The formula includes a crucial parameter ϵ, which controls the probability of maintaining privacy. A smaller ϵ value indicates a higher degree of privacy, as it corresponds to minimal changes in the model's output when a single training sample is altered. Conversely, a larger ϵ suggests lesser privacy. DP is defined by the following inequality:

$$\Pr[\mathcal{M}(x) \in S] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{M}(x') \in S] + \delta, \tag{3.1}$$

where:

- $\Pr[\cdot]$ denotes the probability.
- $\mathcal{M}$ is the mechanism for introducing differential privacy on some dataset x.
- x and x' are two adjacent datasets, differing by at most one element.
- S is the set of all possible outputs of the mechanism $\mathcal{M}$.
- ε (epsilon) is the privacy loss parameter, where smaller values of ε provide stronger privacy guarantees.
- δ is a parameter that accounts for the probability of privacy loss, ideally as close to zero as possible.

The inequality states that the probability of any output from the private mechanism $\mathcal{M}$ on input x being in the set S is bounded by the exponential of ε times the probability of the same output when the input is x', plus a small probability δ. This ensures that the mechanism does not significantly increase the risk of revealing any individual entry in the dataset.

Implementing DP in a federated learning context involves incorporating it during the local training rounds on the client devices, as shown in Figure 3.6. This integration ensures that each client's data contributes to the model's learning without compromising individual data privacy. We observe that integrating DP into machine learning models introduces a trade-off

between privacy, model effectiveness, and computation time. Since DP can increase the time for the model to converge, it is crucial to find an optimal balance that achieves desired privacy levels without significantly compromising the model's effectiveness.

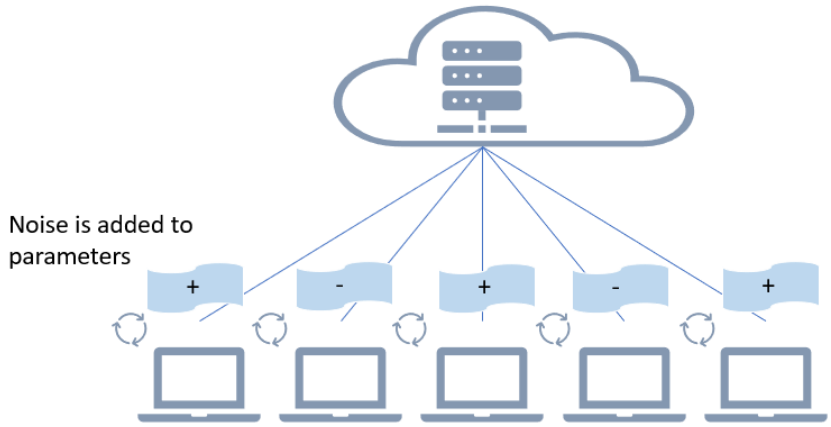


Figure 3.6: Incorporating Differential Privacy in Federated Learning

4 | Problem formulation

A federated learning setup consists of 5 main components:

- Local model M_{local}
- Global model M_{global}
- Client c_i
- Aggregation Server s
- Data D_i

Such setup is a collaborative approach that involves multiple clients, denoted as $\{c_1, c_2, \ldots\}$, each possessing its own unique dataset, represented by D_i for client c_i. The primary objective is to jointly train a global model, denoted as M_{global} that exists on a central server s, by leveraging the knowledge within individual datasets, all while preserving the privacy and security of each client's data.

In the initial phase of federated learning, every client initiates a local model, denoted as M_{local_i}, specifically fitted to its dataset D_i. This local training process allows each client to capture and refine patterns and features inherent to its data:

$$M_{\text{local}_i} = \text{Train}(M_{\text{local}_i}, D_i)$$

Following local training, the updated local models $\{M_{\text{local}_1}, M_{\text{local}_2}, \ldots\}$ undergo an aggregation step on server s to produce the global model M_{global}. This aggregation typically involves averaging the model parameters across all participating clients:

$$M_{\text{global}} = \frac{1}{N} \sum_{i=1}^{N} M_{\text{local}_i}$$

Here, N represents the total number of clients. The updated global model is then distributed back to all clients, where each client replaces its local model with the newly received global

model:

$$M_{\text{local}_i} \leftarrow M_{\text{global}}$$

This iterative process of local training, model aggregation, and global model distribution is repeated to enhance the performance of the global model. As the federated learning algorithm progresses, the global model effectively assimilates knowledge from all participating clients, creating a robust model that reflects the collective intelligence of the decentralized datasets.

The privacy requirements of the clients and aggregation server are similar, both devices require a high level of security utilizing encryption and authentication to prevent an adversary from gaining access to the device itself and as a result, gaining access to the local model and data stored on the client. By gaining access to the local data, the adversary could perform a data poisoning attack. In the situation where one of the client devices is compromised, Additional privacy measures can help to maintain the privacy of local data stored on the other clients. These measures aim to reduce the leakage of useful information by the model. Leaked information can be leveraged by an attacker to determine if a record is part of one of the other clients their local datasets as in an IMIA. This requirement can be fulfilled by making use of DP, as depicted by the equation shown here 3.1.

To define an IMIA, which is simulated in this paper, 3 additional main components need to be formalized. The components are:

- Attack model M_{attack}
- Shadow data D_{shadow}.
- Shadow model $M_{\text{shadow\_}i}$

The objective of an attack model, denoted as M_{attack} is to infer from the global model M_{global} which data records are part of a client c_i their local data D_i . The shadow data, denoted as D_{shadow} is comprised of counterfactual explanations created by using prior domain knowledge, inferred from the model itself or a combination of both.

$$D_{\text{shadow}} = \text{DiCE}(M_{\text{local}_i}, D_i)$$

This data is similar to the adversary client their local data and is used to train a series of shadow models, which are denoted as $\{M_{\text{shadow}_1}, M_{\text{shadow}_2}, \ldots\}$. The shadow models are designed to replicate the behavior of the victim model and are trained using the shadow data.

$$M_{\text{shadow}_i} = \text{Train}(M_{\text{shadow}_i}, D_{\text{shadow}})$$

The attack model aims to differentiate between data points that were part of the original training dataset and those that were not. It is trained using the output of the shadow models, with the objective of determining whether a specific data point was used in training the victim model or not.

$$\hat{y}_{\text{shadow}_i} = \text{Predict}(M_{\text{shadow}_i}, D_{\text{shadow}})$$

$$M_{\text{attack}} = \text{Train}(M_{\text{attack}}, \hat{y}_{\text{shadow}_i})$$

The predictions of the trained attack model classifies if a record is included in the original training data or not. It can be problematic to generate a good-quality shadow dataset without prior domain knowledge. Realizing the importance of this data in an IMIA, and the importance of simulating good IMIA's to determine model resilience, this study aims to investigate the use of counterfactual explenations as the shadow dataset of an adversary in an IMIA. There already exist ways to infer a shadow dataset from the victim model but when talking about an IMIA there is an additional advantage to the adversary. Namely, by gaining access to the client it is possible that also access to the local data is obtained. This local data will be used to provide additional information to the shadow dataset-generating algorithm.

Besides investigating this, performance metrics of a federated setup are compared to a centralized setup, in cases with various levels of DP applied.

5 | Methodology

To easily understand the structure of the experiments and which order they are done, the following Figure 5.1 was created showing all the building blocks. Each block will be further discussed in the subchapters as well as the initialize phase where more information is given about the used data, model hyperparameters, model structure and evaluation metrics.

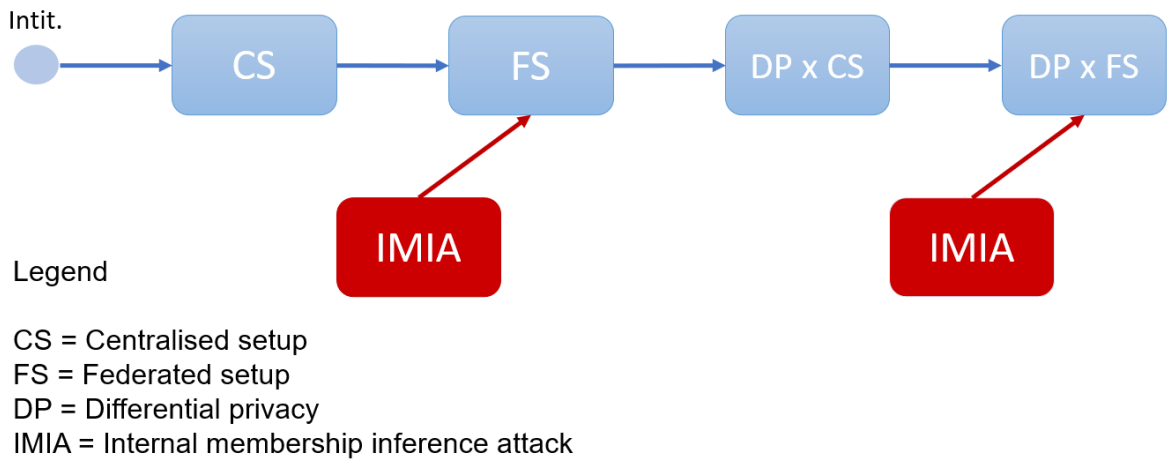


Figure 5.1: Order of experiments

5.1. Baselines

5.1.1. Data

The dataset comes from simulated long-haul networks in a dynamic network operation and contains 32 features to estimate which can be used to predict one of the 4 target variables. One being a class label used to classify if the quality of transmission (QoT) of a network is sufficient or not. The other 3 are continuous variables. In this paper, the class label will be used to make predictions, it is thus a classification task. All the features included in the model can be seen in Table 5.1 below along with their description. Table 5.2 shows the names and descriptions of all the target variables. The data is available on request [27].

Feature names with description	
path_len	Path length [km]
avg_link_len	Average link length [km]
min_link_len	Minimum link length [km]
max_link_len	Maximum link length [km]
num_links	Number of links [1]
num_spans	Number of spans [1]
freq	Carrier frequency [THz]
mod_order	Cardinality of the modulation format [1]
lp_linerate	Lightpath line-rate [Gb/s]
conn_linerate	Connection line-rate [Gb/s]
src_degree	Source node degree [1]
dst_degree	Destination node degree [1]
sum_link_occ	Total number of channels allocated on the links of the lightpath [1].
min_link_occ	Minimum link occupation [1]
max_link_occ	Maximum link occupation [1]
avg_link_occ	Average link occupation [1]
std_link_occ	Standard deviation of the link occupation [1]
max_ber	Maximum BER of lightpaths sharing link(s) with the LUT [1]
min_ber	Minimum BER of lightpaths sharing link(s) with the LUT [1]
avg_ber	Average BER of lightpaths sharing link(s) with the LUT [1]
min_mod_order_left	Minimum modulation order on the channel frequency left to the LUT, considering all links the LUT passes through [1]
max_mod_order_left	Maximum modulation order on the channel frequency left to the LUT, considering all links the LUT passes through [1]
min_mod_order_right	Minimum modulation order on the channel frequency right to the LUT, considering all links the LUT passes through [1]
max_mod_order_right	Maximum modulation order on the channel frequency right to the LUT, considering all links the LUT passes through [1]
min_lp_linerate_left	Minimum lightpath line-rate on the channel frequency left to the LUT, considering all links the LUT passes through [Gb/s]
max_lp_linerate_left	Maximum lightpath line-rate on the channel frequency left to the LUT, considering all links the LUT passes through [Gb/s]
min_lp_linerate_right	Minimum lightpath line-rate on the channel frequency right to the LUT, considering all links the LUT passes through [Gb/s]
max_lp_linerate_right	Maximum lightpath line-rate on the channel frequency right to the LUT, considering all links the LUT passes through [Gb/s]
min_ber_left	Minimum BER on the channel frequency left to the LUT, considering all links the LUT passes through [1]
max_ber_left	Maximum BER on the channel frequency left to the LUT, considering all links the LUT passes through [1]
min_ber_right	Minimum BER on the channel frequency right to the LUT, considering all links the LUT passes through [1]
max_ber_right	Maximum BER on the channel frequency right to the LUT, considering all links the LUT passes through [1]

Table 5.1: Feature information table.

Target variables with description. (source: [27])	
class	Binary class label (0: insufficient QoT, 1: sufficient QoT)
osnr	Optical-signal-to-noise ratio (OSNR) [dB]
snr	Signal-to-noise ratio (SNR) [dB]
ber	Bit-error-rate (BER) [1]

Table 5.2: Target variable information table. (source: [27])

The original data has 1.500.000 synthetic records, but to predict the connection label, 100.000 randomly sampled rows were deemed sufficient, as the data predicts very well the label. This subset was split into 20 percent for testing and the other part for training.

5.1.2. Model

This section describes the model structure and hyperparameters used in the centralized and federated setup. The model in both these setups is exactly the same, to make valid comparisons between them. Likewise, it is also used in the shadow models of an IMIA, since its purpose is to mimic the victim model.

In detail, a shallow multilayer perceptron was chosen with the following structure visible in Figure 5.2. The reason for this network is that the target variable appears to be easily learned from the data, and adding additional complexity to the network did not improve the performance, only increased the running time.

```
Binary_classifier(
  (input): Linear(in_features=32, out_features=64, bias=True)
  (act_input): ReLU()
  (hidden): Linear(in_features=64, out_features=128, bias=True)
  (act_hidden): ReLU()
  (output): Linear(in_features=128, out_features=1, bias=True)
  (act_output): Sigmoid()
)
```

Figure 5.2: Multilayer perceptron structure

The model contains 3 linear layers and uses a sigmoid activation function for the output, so that the result is compressed to either 0 or 1.

As an optimizer, stochastic gradient descent (SGD) is used. The model is trained until convergence, which occurs at a varying number of training rounds depending on the used setup and what level of privacy measures are applied. Training is done with a batch size of 512 in the centralized setup and 128 in the federated setup, using a learning rate of 0.01 and momentum of 0.9. Since it is a binary classifier, as criterion binary cross entropy was chosen.

5.1.3. Evaluation metrics

Since from the dataset the binary connection quality label was chosen as the target variable, the model is a classifier. To evaluate the performance of this classifier 4 metrics were used:

- Accuracy : $\frac{\text{True positives + True Negatives}}{\text{Total instances}}$
- Precision : $\frac{\text{True positives}}{\text{True positives + False positives}}$
- Recall : $\frac{\text{True positives}}{\text{True positives + False negatives}}$
- F1 score : $2 * \frac{\text{Precision * Recall}}{\text{Precision * Recall}}$

Negative meaning the prediction of a bad connection network, while positive would be a good connection network. The focus will mainly be on accuracy and F1 since F1 incorporates both precision and recall by using their harmonic mean.

In the federated setup, there are two ways evaluation can be done. Either on a specific client using only a subset of its local data as a test set. Or on the server side, using the aggregated metrics from all other participating clients. It was chosen to use server-side evaluation since it offers a better overview of the general performance of the model. After each global training round, when the clients send the model parameters to the server, another object is also sent including all the metrics, computed on each client their local test set. On the server these metrics are aggregated using the average to obtain the model performance of that global training round.

5.2. Centralized setup

For the centralized model, all of its 100.000 rows of data were available on the device itself, and the model was trained and evaluated using the methods described in Section 5.1.3.

DP is applied to the centralized model during the training process by adding structured noise to the gradient updates. In the experiments, privacy guarantee ε (epsilon) levels of 100, 5 and 1 were used. The noise-generating mechanism $\mathcal{M}$ is a cryptographically safe pseudo-random number generator. The δ (delta) parameter was set to 0.00001 as the best practice for the parameter is to set it equal or smaller than the inverse length of the train dataset, which in this case equals to 80.000 rows.

5.3. Federated setup

In the federated setup, the data of 100.000 rows was distributed equally over 5 clients by random sampling. Each client has as local data 16.000 rows for training and 4000 for testing.

Since all the clients have the same features in the data, it is a horizontally split federated learning setup. The clients were simulated on a local network and linked to the server address, which also exists on the local network. On each client, the model structure and its hyper-parameters are defined, while on the server, the aggregation strategy for the model parameters and for the performance metrics are defined.

For each global training round, the clients use 10 rounds of training on their local data. After all local training rounds are completed the computed model updates are sent to the server where they are aggregated using the average. All clients participate in every aggregation round, so there exists no selective aggregation process. In the network, all the clients have the same bandwidth and there are no clients losing connection. The amount of global training rounds used depends on how fast the model converges. In the federated setup used in this paper, the network connection dataset was split into 5 equal parts of 20.000 rows. So that each of the 5 clients has a unique subset of equal length with the same features. For that reason, it is a horizontally split federated learning setup.

When applying DP to this setup, the noise is added during the training process on each client. The same settings for the DP algorithm were used as for the centralized setup.

5.4. Internal membership inference attack

In this paper, MIAs were applied internally to a federated setup, the scenario where an adversary was able to gain access to one of the clients. The IMIA was done using 6 shadow models and one attack model. The shadow models use the same structure as the victim model. When varying the number of shadow models, the attack performance did not increase much only the training time so it was chosen to stick with 6. The shadow models were trained until convergence and for their training data, 3 shadow datasets were inferred from the victim model with ε (epsilon) levels of 100, 5 and 1 applied.

Various settings of ε in DP were used to assess its effectiveness in constraining the extent to which precise data can be inferred from the model. The inference of data was conducted utilizing the DiCE algorithm as described in Section 3.4 and depicted in Figure 5.3.

The DiCE method was chosen as it is assumed the adversary has no previous knowledge about the data, so is required to infer the data only from the model. Specifically, counterfactual explanations were chosen as the attacker has access to the client their local data, which can be used as feature vectors to input into the DiCE algorithm.

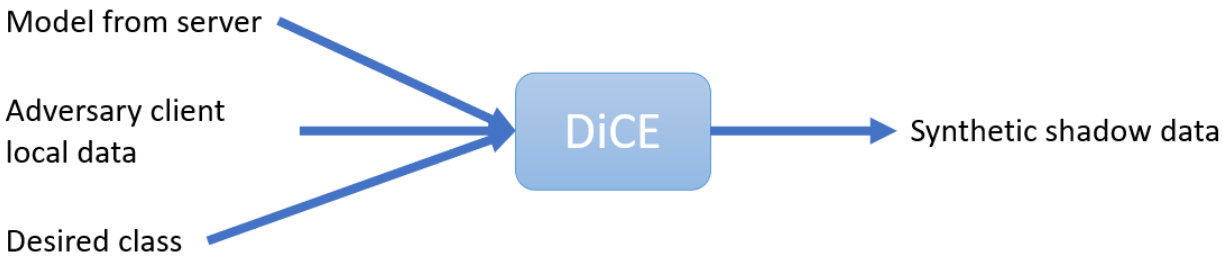


Figure 5.3: DiCE applied to generate shadow data for use in an IMIA

Besides feature vectors from the client their local data, DiCE was provided with the trained model from the server and a desired output, which was the opposite of the target class present in the compromised client their local data. The attack is created from one client and tries to predict which records are part of another client their local data. This attack was done for all clients and to evaluate the performance, local data from the other victim clients was used.

6 Results

This section presents the findings obtained from the experiments outlined in the methodology. Firstly, an examination of the performance differences between centralized and federated models under different DP settings is conducted. Subsequently, the simulation of an IMIA is performed on federated models, each with a different DP setting to evaluate the influence of the ε (epsilon) parameter on the attack its performance. Finally, the DiCE algorithm is used in an attempt to infer data from a model trained on images.

6.1. Centralized and federated model performance comparison

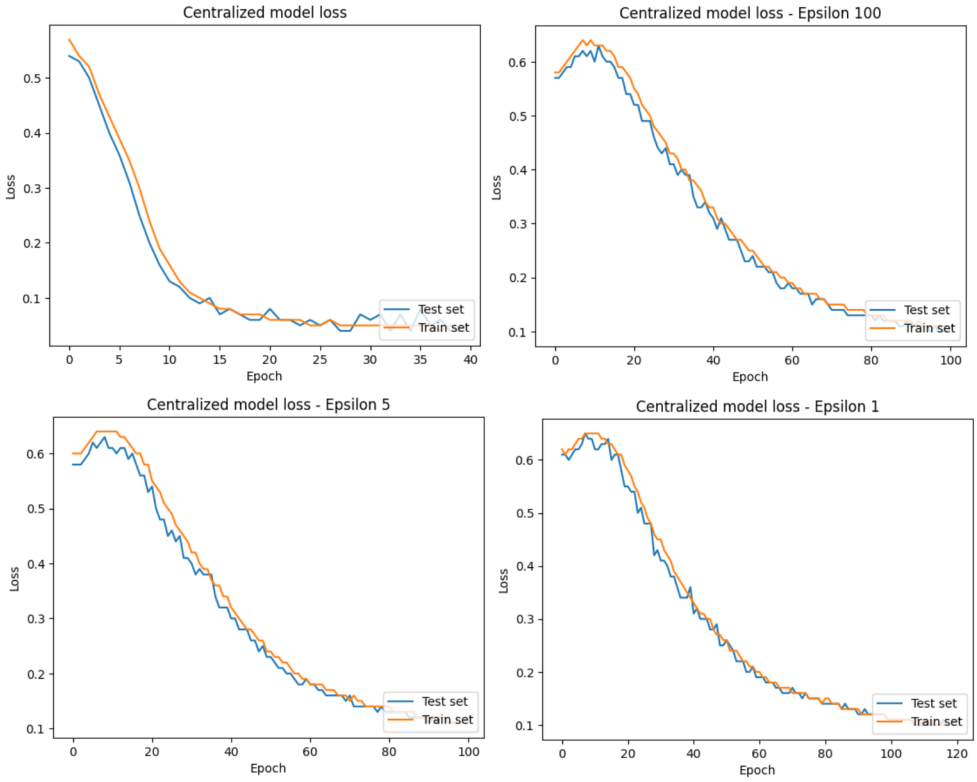


Figure 6.1: Centralized model loss comparison with various levels of DP applied

In Figure 6.1 above, it can be seen that the centralized model converges slower when lowering the ε (epsilon) parameter of the DP formula. Without DP, the model converges at around 40 epochs but with DP applied and ε = 1, convergence takes 4 times longer. Also, it can be seen that when DP is applied, for the first 10 epochs, the loss increases.

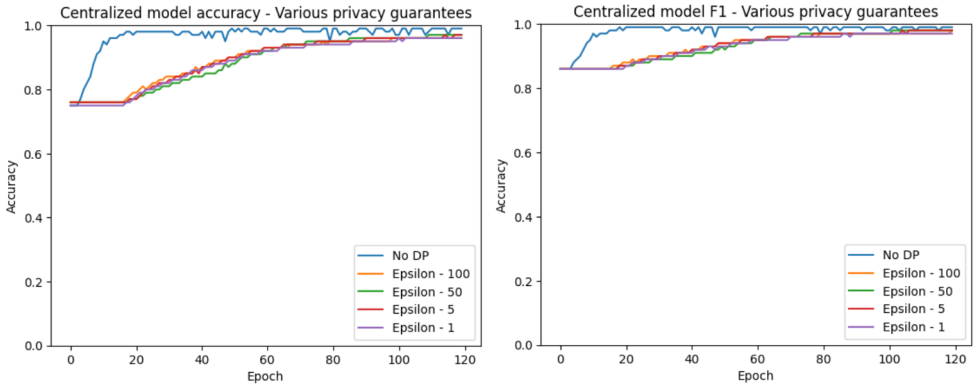


Figure 6.2: Accuracy and F1 score of centralized models with varying levels of DP

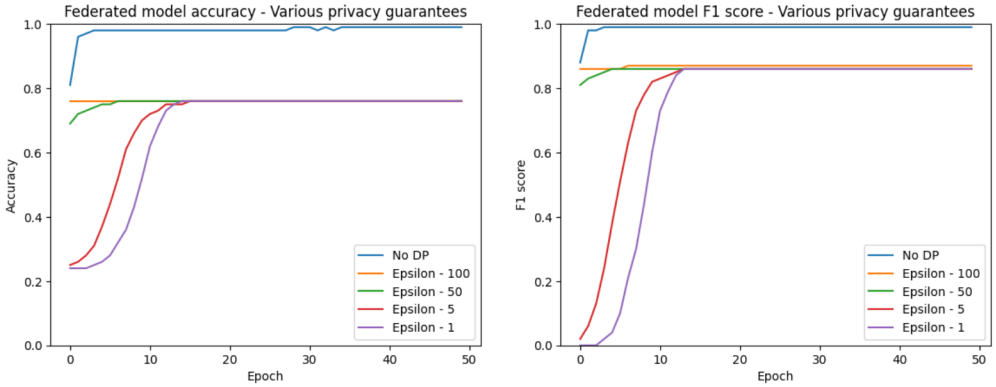


Figure 6.3: Accuracy and F1 score of federated models with varying levels of DP

In Figures 6.2 and 6.3 above, F1 scores and accuracies are shown from both centralized and federated setups. it can be seen that without DP applied, both the federated and centralized models achieve very similar performance metrics. But when applying DP, the federated model eventually stabilizes at an accuracy of 0.76 and an F1 of 0.86 for all ε settings. In contrast to the centralized model which achieves an accuracy of 0.95 and a F1 score of 0.98 even with DP applied. In both centralized and federated setups, as the value of ε decreases, the model requires more training rounds to reach its optimal performance. Moreover, in the federated setup, the amount of epochs required exhibits exponential growth as ε decreases. In that setup, there is minimal difference in the effect between ε of 50 and 100, but a more pronounced effect is observed with an ε of 5. This observation aligns with the non-linear relationship between ε and the privacy guarantee.

It can also be noted that the performance metrics of the federated setup without DP increase faster compared to the centralized setup. This can be explained by the fact that for every global epoch, in the federated setup, the model already has 10 client-side epochs. In a real-life scenario, the performance of the federated setup would be lower since often the networks have a large number of clients and there is a probability that certain clients drop out during the process when, for example internet connection is lost. As a result their local model parameters are not included, which in turn affects the global model performance. Also, in a more realistic scenario, more time would be required for the server to aggregate all the model updates, since the server needs to wait in each global training round for a certain amount of clients to have submitted their updates. This time can be delayed by varying network bandwidth and computational resources on the client device.

Besides using the average as an aggregation strategy, an experiment using the median was done, however, there was no notable change in performance. Using the average or median as in this paper is acceptable since all clients have an equal number of data records with similar distributions. But in a scenario where the data is unequally distributed among the clients, using the average as an aggregation strategy can mean losing important information as some clients their model updates are more meaningful than others.

In Table 6.1 below, the exact performance numbers can be seen from the previously described experiments. From Table 6.1 it appears that DP does not really impact the performance but as shown in the previous Figures 6.1, 6.2 and 6.3, in most cases the main difference is that more training rounds are required to achieve the same performance.

Performance metrics				
Experiment	Accuracy	Recall	precision	F1
CS	0.99	0.99	0.99	0.99
CS - DP ε 100	0.96	0.99	0.99	0.98
CS - DP ε 50	0.97	0.99	0.99	0.98
CS - DP ε 5	0.97	0.99	0.99	0.98
CS - DP ε 1	0.96	0.99	0.99	0.98
FS	0.99	0.99	0.99	0.99
FS - DP ε 100	0.76	0.99	0.76	0.87
FS - DP ε 50	0.76	0.99	0.76	0.86
FS - DP ε 5	0.76	0.99	0.76	0.86
FS - DP ε 1	0.76	0.99	0.76	0.86

Table 6.1: Exact metrics from all experiments.

Table 6.1 above illustrates that in all experiments the recall value is very high at 0.99. This means that the model predicts almost all records with good connection correctly. Similar values of 0.99 and 0.98 can be observed for the precision and F1 score, but only in the centralized setup and the federated setup without DP. Across federated setups with DP, the precision is approximately 20 percent lower compared to the precision observed in other

experiments. The lower precision indicates that the model tends to make more false positive predictions, more specifically for the model this means misclassifying instances with an actual class of 0 (bad connection) as belonging to class 1 (good connection). This can be seen also in the Table 6.2 below which shows for each class the amount of predicted values. As a result, the accuracy is approximately 20 percent lower since the high recall is offset by the lower precision. When applying DP to the centralized setup, the accuracy also decreases by around 0.03.

Predictions class distribution original model			
classes	test set	prediction with ε 100	prediction with ε 1
0	4893	4559	263
1	15107	15441	19737

Table 6.2: Distribution of class prediction from original model with different epsilons.

6.2. Internal membership inference attack with various privacy guarantees

For the next experiments, 4 attacks were launched from client 1 to every other client with the original model not using DP. As can be seen from Table 6.3 below, the performance of the attack is very similar on all clients. This is as expected because in the simulation, all of the clients use subsets of the same data as their local data which are randomly sampled and of equal size.

Performance of internal attack model without differential privacy				
Client	Accuracy	Recall	precision	F1
2	0.60	0.77	0.67	0.71
3	0.59	0.78	0.67	0.72
4	0.58	0.75	0.66	0.71
5	0.58	0.76	0.66	0.71

Table 6.3: Internal membership inference attack from client 1 to all other clients without differential privacy.

As the result is very similar for all clients, in the next experiment the attack was just done on one client with various DP settings. In Table 6.4 below it can be seen that DP indeed makes a significant impact on the performance of an attack. When using ε 1, the attack loses 35 percent in accuracy compared to when no DP is used. With any of the 3 privacy settings, the model becomes useless because the accuracy drops below 50 percent and thus being worse than random guessing.

The other metrics also decrease significantly when ε equals 1, the change is in line with the decrease seen in the accuracy. When increasing the amount of training rounds on the shadow models, the attack does not manage to exceed the performance score listed in Table 6.4.

Performance of internal attack model with differential privacy				
Epsilon	Accuracy	Recall	precision	F1
100	0.45	0.61	0.46	0.53
5	0.38	0.52	0.40	0.45
1	0.26	0.39	0.29	0.33

Table 6.4: Internal membership inference attack model performance with differential privacy.

7 Conclusion

DP can greatly improve the resistance against IMIA's since it forces to model to learn patterns, instead of specific data. this resilience is proven in the previous Section 6.2, but it keeps being a trade-off between performance and privacy. The federated model used in this paper achieved a very high performance easily so the effect of applying DP with a very low epsilon parameter still results in an acceptable model. But, often on real data the model performances are much lower and then, this performance decrease could be unacceptable. Further, it was shown that a DiCE algorithm is a possible way to infer shadow data from a victim model in an internal membership inference attack, since the malicious client their local data can be used as query vectors for the DiCE algorithm. Although, the quality of such a shadow dataset greatly depends on the data-dimensions the victim model is trained on, as discussed in the Section 3.4.

At this moment, the amount of up-to-date libraries offering federate learning schemes and especially the libraries to simulate attacks such as an MIA are still limited. However, the field is getting an increasing amount of attention, especially with the recent updates in legislation around privacy-oriented and secure AI models in various countries around the world.

Further work could be done to integrate homomorphic encryption into the federated setup, possibly combining it with DP and to test out the impact on IMIA's as well as on other attacks such as model inversion. Additionally, a more advanced federated setup could be used with selective aggregation and unequally distributed data.

Bibliography

[1] European Commission, Directorate-General for Justice and Consumers, The GDPR – New opportunities, new obligations – What every business needs to know about the EU's General Data Protection Regulation, Publications Office (2018) https://data.europa.eu/doi/10.2838/97649

[2] Council of the EU (2023). Artificial intelligence act: Council and Parliament strike a deal on the first rules for AI in the world https://www.consilium.europa.eu/en/press/press-releases/2023/12/09/artificial-intelligence-act-council-and-parliament-strike-a-deal-on-the-first-worldwide-rules-for-ai/

[3] Assembleia Legislativa do estado de sao paulo (2020). LEI GERAL DE PROTEÇÃO DE DADOS https://www3.al.sp.gov.br/repositorio/arquivoWeb/cgp/informativos/cgp9292.pdf

[4] European data protection board (2023). 1.2 billion euro fine for Facebook as a result of EDPB binding decision https://edpb.europa.eu/news/news/2023/12-billion-euro-fine-facebook-result-edpb-binding-decision$_en$

[5] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, Peter Richtárik (2016). Federated Optimization: Distributed Machine Learning for On-Device Intelligence https://arxiv.org/abs/1610.02527

[6] Rodolfo Stoffel Antunes, Cristiano André da Costa, Arne Küderle, Imrana Abdullahi Yari, Björn Eskofier (2022). Federated Learning for Healthcare: Systematic Review and Architecture Proposal https://dl.acm.org/doi/abs/10.1145/3501813

[7] Deep Kawa1, Sunaina Punyani1, Priya Nayak1, Arpita Karkera1, Varshapriya Jyotinagar2 (2019). Credit Risk Assessment from Combined Bank Records using Federated

Learning https://www.academia.edu/download/59805474/IRJET-V6I428820190620-14769-hvp12k.pdf

[8] Yong Song, Yuchen Xie, Hongwei Zhang, Yuxin Liang, Xiaozhou Ye, Aidong Yang, Ye Ouyang. (2021). Federated Learning Application on Telecommunication-Joint Healthcare Recommendation https://ieeexplore.ieee.org/document/9657870

[9] Fei Tang, Shikai Liang, Guowei Ling, Jinyong Shan (2023). IHVFL: a privacy-enhanced intention-hiding vertical federated learning framework for medical data https://link.springer.com/article/10.1186/s42400-023-00166-9

[10] Yusuf Efe (2021). A Vertical Federated Learning Method For Multi-Institutional Credit Scoring: MICS https://arxiv.org/abs/2111.09038

[11] Solmaz Niknam, Harpreet S. Dhillon, Jeffrey H. Reed (2021). Federated Learning for Wireless Communications: Motivation, Opportunities, and Challenges https://ieeexplore.ieee.org/abstract/document/9141214

[12] Xingchen Zhou, Ming Xu,Yiming Wu, Ning Zheng (2021). Deep Model Poisoning Attack on Federated Learning https://www.mdpi.com/1999-5903/13/3/73

[13] Haotian Liang, Youqi Li, Chuan Zhang, Ximeng Liu, Liehuang Zhu (2020). EGIA: An External Gradient Inversion Attack in Federated Learning https://ieeexplore.ieee.org/abstract/document/10209197

[14] Jingwen Zhang, Jiale Zhang, Junjun Chen, Shui Yu (2020). GAN Enhanced Membership Inference: A Passive Local Attack in Federated Learning https://ieeexplore.ieee.org/abstract/document/9148790

[15] Priyanka Mary Mammen (2021). Federated Learning: Opportunities and Challenges https://arxiv.org/abs/2101.05428

[16] HONEUR, (2023). Accelerated real-world data analysis and evidence sharing https://www.janssen.com/emea/janssen-signs-6-more-partners-honeur-project;

https://portal.honeur.org/

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim Adam Smith (2006). Calibrating Noise to Sensitivity in Private Data Analysis https://link.springer.com/chapter/10.1007/11681878$_14$

[18] Alizishaan Anwar Hussein Khatri (2017). Preventing Overfitting in Deep Learning Using Differential Privacy https://www.proquest.com/openview/395ac461a83872e0772f3a65a7dc5060/1?pq-origsite=gscholarcbl=18750

[19] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, H. Vincent Poor (2019). Federated Learning with Differential Privacy: Algorithms and Performance Analysis https://arxiv.org/abs/1911.00222

[20] R. Shokri, M. Stronati, C. Song and V. Shmatikov (2017). Membership Inference Attacks Against Machine Learning Models https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=7958568

[21] Cynthia Dwork, Aaron Roth (2014). The Algorithmic Foundations of Differential Privacy https://www.cis.upenn.edu/ aaroth/Papers/privacybook.pdf

[22] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, H. Brendan McMahan (2021). Adaptive Federated Optimization of Differential Privacy https://arxiv.org/abs/2003.00295

[23] Reza Shokri, Marco Stronati, Congzheng Song, Vitaly Shmatikov (2017). Membership Inference Attacks against Machine Learning Models https://arxiv.org/abs/1610.05820

[24] Joon-Woo Lee, Hyungchul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee. (2022). Privacy-Preserving Machine Learning With Fully Homomorphic Encryption for Deep Neural Network https://ieeexplore.ieee.org/abstract/document/9734024

[25] Henger Li, Xiaolin Sun, Zizhan Zheng (2022). Learning to Attack Federated Learning: A Model-based Reinforcement Learning Attack Framework

https://proceedings.neurips.cc/paper$_files/paper/2022/hash/e2ef0cae667dbe9bfdbcaed1bd91807b-Abstract-Conference.html$

[26] Kathrin Melcher (2021). A Friendly Introduction to (Deep) Neural Networks https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks

[27] Geronimo Bergk, Behnam Shariati, Pooyan Safari, Johannes K. Fischer (2021). ML-assisted QoT estimation: a dataset collection and data visualization for dataset quality evaluation https://www.hhi.fraunhofer.de/en/departments/pn/products-and-solutions/qot-dataset-collection.html

[28] Sagar Sharma (2017). Activation Functions in Neural Networks https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[29] Ana Lucic, Harrie Oosterhuis, Hinda Haned, Maarten de Rijke. (2021). FOCUS: Flexible Optimizable Counterfactual Explanations for Tree Ensembles https://arxiv.org/abs/1911.12199

[30] Jing Li, Ji-hang Cheng, Jing-yuan Shi, Fei Huang (2015). Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement https://link.springer.com/chapter/10.1007/978-3-642-30223-7$_87$

[31] Hong Hui Tan, King Hann Lim (2019). Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization https://ieeexplore.ieee.org/abstract/document/8843652

[32] Ramaravind Kommiya Mothilal, Amit Sharma, Chenhao Tan (2019). Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations https://arxiv.org/abs/1905.07697

[33] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Sha Wei, Fan Wu, Guihai Chen, Thilina Ranbaduge (2022). Vertical Federated Learning: Challenges, Methodologies and Experiments https://arxiv.org/abs/2202.04309

[34] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, Ilya Mironov (2022). Opacus: User-Friendly Differential Privacy Library in PyTorch https://arxiv.org/abs/2109.12298

[35] Yann LeCun, Corinna Cortes (2005). The MNIST Dataset Of Handwritten Digits (Images)

http://www.pymvpa.org/datadb/mnist.html

[36] Sander Peeters. (2024). Internal membership inference attack on private federated models. Github. https://github.com/SanderPe/IMIA-on-federated-private-models

A | Appendix

On the following GitHub repository [36], the pipeline created for conducting all the experiments can be found.

List of Figures

List of Tables

,

,

,

Acronyms

DiCE Diverse Counterfactual Explanations

DP Differential Privacy

GDPR General Data Protection Regulation

IMIA Internal Membership Inference Attack

MIA Membership Inference Attack

Acknowledgements

I want to thank Ezzeddine Fatima and Davide Andreoletti for the precise feedback and guidance during this project.