Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Assessing and selecting ϵ for differentially private Federated Learning with Inference Attacks

Tom Ganz

Roll number	71127
External Supervisor	Daniel Bernau (SAP SE)
Masters Committee	Prof. Dr.-Ing. Astrid Laubenheimer
Department	Computer Science and Business Information System

Abstract

Training federated Machine Learning models is an emerging trend: multiple data owners train an identical local model and send their model parameters to an aggregator. In this protocol, the local models with identical structure over their respective local datasets, get aggregated and sent back to the data owners. Although, the data owners do not send their potentially sensitive data to the other participants or the aggregator, it is still possible, to extract identify the presence of specific attributes or records in the local training data from each data owner during the collaborative training. To improve the privacy guarantees, ϵ-Differential Privacy can be applied within the local data owner premises. Either as Local Differential Privacy, where the training data is perturbed or using Central Differential Privacy by perturbing the gradients during training. Balancing privacy and model utility is not a trivial task and the privacy parameter ϵ needs to be fine-tuned accordingly. In this thesis, inference attacks are used to asses the privacy parameter of Machine Learning models in Federated Learning for some exemplary datasets. This thesis aims to formulate suggestions for data scientists and data owners for choosing ϵ and comparing suitable privacy-utility trade-offs. In this work, a Federated Learning framework, a white-box Membership Inference Attack Model and a novel Attribute Inference Attack Model are introduced. The results of this thesis show, that Central Differential Privacy is favorable over Local Differential Privacy and that an adversarial Federated Learning central instance constitutes a stronger position for an inference attack than an adversarial participant in the training. Furthermore, this work shows, that already a high ϵ is sufficient to mitigate most attacks.

Contents

Introduction

1.1 Motivation

Machine learning algorithms based on neural networks are used in various domains. Often these models require a large representative amount of data to be effective. These datasets may be crowdsourced and may contain personal data or sensitive business information [1]. Personal data could consist of photos, voice recordings or medical information. Additionally such personal or sensitive business data cannot be easily removed from a learned model [2]. For above reasons, it is crucial as a company to protect the confidentiality of the training data with privacy enhancing technologies.
Commonly a machine learning model is trained by a single data analyst on a central server. However, due to the increasing amount of storage and computational power of mobile and embedded devices, it seems promising to train data locally. This enables the possibility to train models collaboratively, by first training locally and then centrally aggregating the individual models [3]. The gain of this so called "federated learning" is scalability on one hand, due to the fact that multiple devices train independently on their hardware and in consequence privacy on the other, since a possibly untrusted server does not see the actual training data from the clients [4]. In theory the latter is enforced by keeping the sovereignty of the sensitive or personal training data by its respective data owner.
However previous research has shown that without much knowledge of the actual training data, a honest-but-curious attacker might infer individual training data records or attributes from a trained deep learning model [5]. Thus a data analyst acting as the aggregator is capable of inferring sensitive information from the model itself. There are several methods to assure privacy of the training data [6]. In the thesis one method called "differential privacy" (DP) is further investigated [7]. This technique provides a trade-off

metric between privacy guarantee and model usability. DP provides mechanisms like perturbing the training data or an aggregation function to enforce privacy [5]. However choosing correct DP parameters and mechanisms is a non-trivial task especially for data scientists not familiar with differential privacy. Using threat models such as "membership inference" enables the possibility to evaluate different privacy strategies for their actual effectiveness.
In this thesis differentially private models will be trained using personal or sensitive training data and federated learning algorithms. Differential privacy is used to secure the sensitive data. Depending on the accuracy of inference attacks the parameters of the differential private mechanisms can be evaluated. The purpose of this thesis is to derive a suggestion for the usage of parameters and mechanisms to ensure differential privacy for federated learning.

1.2 Environment

The thesis will be created at "SAP SE" in the SAP Security Research department. SAP is the world's leading provider of business software solutions. SAP solutions help enterprises of all sizes around the world to improve customer relationships, enhance partner collaboration and create efficiencies across their supply chains and business operations. SAP group includes subsidiaries in over 180 countries and employs more than 100.000 people.
Taken from the Security Research strategy description, the research focuses are, amongst other relevant topics, anonymization approaches based on differential privacy.

> "SAP Security Research works on novel anonymization approaches based on differential privacy. We aim to make the results of these research activities in anonymization cloud services available for the Intelligent Enterprise as microservices. Meant to be used in all cloud solutions, these microservices will enable the ethical use of AI in an intelligent enterprise" [8, p. 7]

As stated their work consists of scientific research and prototyping. Eventually the results from the research should be applied to real use-cases for customers in e.g. cloud and microservice solutions.
Today the Security Research team consists of four Post-Docs and three PhD-Students.

1.3 Research Question

The promising attack results from Nasr et al. [9] against federated learning models suggest that the model parameters sent by the participants to the aggregator leak enough information to successfully infer sensitive information about individuals in the trained datasets.
In this thesis different state-of-the-art models should be trained with differential privacy on different datasets in a federated fashion. This work should examine the privacy-utility trade-off for these models before and after applying differential privacy mechanisms. Eventually, meaningful privacy parameters for differential privacy should be validated by using inference attacks.

1.4 Methodological Approach

Nasr et al. introduced a white-box membership inference attack model [9]. An adversary with knowledge of the target model parameters can use their work to successfully infer sensitive data. Nasr et al. even show, that its possible to infer sensitive training data in federated learning.
Using "DP" is a non-trivial task, especially for data scientists not familiar with the anonymization techniques. Commonly the choice of ϵ depends on a utility and privacy trade-off. This work examines the attack using Membership Inference on models used by Nasr et al. and Bernau et al. [5]. The goal is to see, if it is possible to give dataset independent suggestions for parameter ϵ with the use of Membership Inference attacks. Although this work focuses on models trained using a federated learning approach, therefor the recommended parameter combination have to fit with the federated learning hyperparameters.
Besides, in this work, it is shown that the membership inference attack model from Nasr et al. can be easily extended to perform an attribute inference attack.

1.5 Structure

Chapter *Theoretical Foundation 2* serves as an introduction to this thesis' building blocks as well as to the related works. After this chapter the terminology should be comfortable for the reader. Chapter *Methodology 3* introduces the experiment setting consisting of hypothesis, implementation, threat model and used metrics. Chapter *Experiments 4* presents results for certain experiments with different datasets. At last, chapter *Discussion 5*

puts the results into perspective.

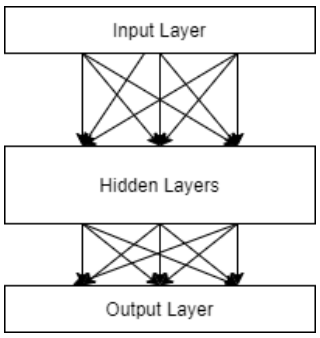
Theoretical Foundation

This chapter serves introduces the research foundations which this work is built upon. First Machine Learning and Deep Neural Networks, afterwards Federated Learning and finally algorithms for evaluation are laid out. At last the main works focusing on Differential Privacy and Membership Inference are introduced.

2.1 Deep Neural Networks

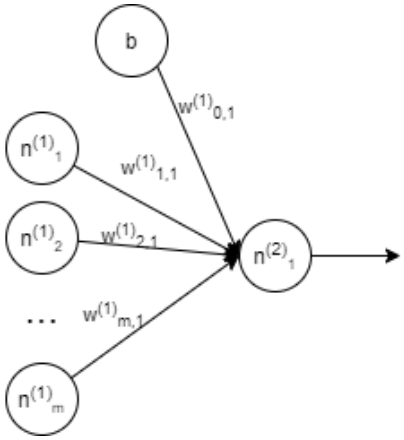
Machine Learning (ML) is a discipline arising from computer science and statistics. The intention of ML is to gain new knowledge by finding recurring patterns in datasets without or with only minimal use of human interaction. Machine Learning is especially used for complex data and therefore replaces traditional statistical methods. Most often problems requiring the use of ML are separated in classification tasks where a model needs to classify a given data point to a given class or perform regression tasks where a model predicts numerical values.
Especially *Artificial Neural Network* (ANN) gained large popularity for this kind of tasks and is a widely adopted Machine Learning concept [10]. An ANN can be seen as an universal function approximator [11].
The architecture of an ANN can be summarized by three components: The neurons (units), the layers and weighted connections between the neurons. Every layer consists of a fixed amount of neurons. All *Deep Neural Networks* (DNN) have multiple hidden layers and the ability to learn complex data representations in common. Each hidden layer learns a specific abstraction of representations from the given dataset. The input layer denotes the dimension of a data point and the output layer denotes the dimension of the desired prediction vector (i.e. the result).

Figure 2.1: ANN Architecture



As seen in Figure 2.1, the input layer is connected to a configuration dependent number of hidden layers. The last hidden layer is connected to the output layer. The output layer yields the predicted result.
Every layer consists of an arbitrary, but fixed amount of neurons as part of their configuration. This can be seen in Figure 2.2.

Figure 2.2: ANN layer configuration.



Let $n_j^{(i)}$ denote the j-th neuron of the i-th layer. Every node $n_j^{(i)}$ is connected to every node $n_k^{(i+1)}$ by an edge with weight $w_{j,k}^{(i)}$. Hence the activation value for a specific node is given by:

$$n_k^{(i+1)} = h\left(\sum_{j=0}^{n} w_{j,k}^{(i)} \cdot n_j^{(i)}\right)$$

The function $h(x)$ specifies a non-linear activation function e.g. *Rectifying Linear Unit* (ReLu) or *Tangens Hyperbolicus* (tanh). The output of such

a function $h(x)$ is called *activation value*. Activation functions are used in order to add non-linear properties to the learning ability of a model. Without these, only linear mappings could be learned.

Forward Pass

Given a data point x, typically denoted as $\vec{x} = [x_0 \dots x_m]$, x is passed through the first layer, in other words: $n^{(0)} = x$. The prediction output of the ANN is given by consecutively passing the values of the neurons to the next layer by calculating the neurons activation values with the calculation rule seen in figure 2.2. In a classification task, i.e. calculating an association between a data point and a label, the output layer commonly returns a vector with the size equal to the number of classes. To obtain a confidence vector typically the output vector is normalized by applying the *Softmax* Function 2.1 on the output values.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.1}$$

If the output value is a single scalar, the *Sigmoid* Function 2.2 is preferred to map a value to a probability.

$$\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{2.2}$$

Gradient Descent

Gradient Descent (GD) is an optimization algorithm that finds minima (the steepest descent) in a function. In this context Gradient Descent is used to fit the parameters of a neural network to a suiting solution by minimizing a loss function. This process is also called *backpropagation*. The algorithm calculates the partial derivatives of a loss function $E : \Omega \to R$ concerning its weights $\frac{\delta J}{\delta W}$. A partial derivative is called gradient, hence the name Gradient Descent. Gradient Descent differentiates, with respect to each parameter, the loss function J and then calculates the error for each sample $d \in D_{train}$. The calculated gradients are then used to take an update for each parameter. The calculated gradients are used to update the weights by:

$$W^{(t+1)} = W^{(t)} - \eta \nabla J(f(x; W^{(t)}), y)$$

The step size and therefore the convergence speed is controlled by a hyper-parameter η. As a major disadvantage, Gradient Descent needs to iterate over the entire dataset before taking a step. This leads effectively to only one

update per epoch. Comparatively the *Stochastic Gradient Descent* (SGD) algorithm, calculates an update for each calculated loss, i.e. for each sample. Each data point $d \in D_{train}$ used during training influences the weights of an ANN. The SGD attempts to minimize the contribution to the training loss for each d [2].

This consequently leads to the fact, that gradients of the loss for data points $d \in D_{train}$ lean to zero, whereas the loss for a data point $d' \notin D_{train}$ will be relatively large. Other variations of GD like *Batch Gradient Descent* calculate the loss for a small subsample of D_{train} and then average over the gradients to update the weights. Variations of SGD, like *Adam*, *AdaGrad*, *RMSProb* tend to converge faster or more stable.

2.2 Federated Learning

Commonly, data scientists collect heterogeneous data and train ML models centrally, assuming for example the popular CRISP-DM process [5]. Typically, during the CRISP-DM process, a data scientist gathers and collects data and afterwards elects a model which is evaluated and eventually deployed. *Federated Learning* (FL), however, distributes the training among different *Data Owners* ($\mathcal{DO}$) [4]. In theory this enforces anonymity, because a data scientist, i.e. the aggregator, does not see the actual potentially sensitive data and therefore the data is kept by the sovereignty of the $\mathcal{DO}$s. The second reason for using FL is a significant performance gain because the computational effort is distributed across the $\mathcal{DO}$'s. By Amdahls law [12] the optimization is archived by parallelizing a sequential algorithm. In fact, the interest in FL techniques arose due to increasing computational and storage resources on handhelds (e.g. mobile phones) and embedded devices.
Federated Learning is similar to *Map-Reduce* and can be better explained with following analogy: The dataset is split (mapped) to independent machines. There, each machine performs a forward- and backpropagation. Finally, a central instance aggregates (reduces) the weights to obtain a central model.
In practice each Data Owner trains a *local model* and sends their parameters, e.g. the respective gradients or weights, to the central aggregator. The aggregator keeps a central model, which gets updated by the aggregated parameters received from each Data Owner, hence the name *aggregator*. The $\mathcal{DO}$ uses the current local model, while the new global model has not been propagated yet. This implies (compared to CRISP-DM), that in FL, there is no single deployment, rather the models are being in use constantly.

One real-world example is the word suggestion used by Google which is based on an ML model learned distributed across several mobile phones [13]. By typing a word, the Android mobile keyboard immediately suggests subsequent words.
Obviously, the data is both: sensitive and not respective for the whole distribution because every person has other writing habits.
Several algorithms can be used to train a model using Federated Learning. In this work, the FedAVG algorithm is further examined.
Agarwal et al. formalize a distributed SGD algorithm. Let $F(w) : \mathcal{R}^d \longrightarrow \mathcal{R}$ be of the form

$$F(w_k) = \frac{1}{M} \sum_{i=0}^{M} f_k(W)$$

where each $f_k(x; W)$ resides at the k-th client [14].
Brendan et al. suggest following algorithm (FedAVG) as seen in Algorithm. 1 [13]. For clarity McMahan et al. define a round, also referred as communication period, as one iteration where all Data Owner train their local models for E epochs [4]. Hyperparameter C defines the fraction of clients that train in parallel for one round. E.g. $C = 0.5$ means that one half of the available clients (K) are instructed to train. Parameter B defines the local batch size.

Algorithm 1 FedAVG

procedure FEDAVG ▷ Run by the server
  initialize w_0
  for each round $t = 0, 1, 2, ..$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m Clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow \text{Clientupdate}(k, w_t)$
    end for
    $w_{t+1} \leftarrow \sum_{k=0}^{K} \frac{n_k}{n} w_{t+1}^k$
  end for
end procedure
procedure CLIENTUPDATE(k, w) ▷ Run by the clients
  $\mathcal{B} \leftarrow$ (split P_k into batches of size B)
  for each local epoch i from 0 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \Delta J(w; b)$
    end for
  end for
  return w to server
end procedure

Algorithm 1 shows, that the server and client communication is asynchronous. The server instructs the clients to run a gradient descent optimization on their local datasets. Afterwards all weights from the local model are collected and averaged to receive the new weight matrix. The averaging takes the training size per Data Owner into account, thus, a Data Owner who trained on a large dataset has therefore much larger impact on the gradients.
Recent work combines Federated Learning with privacy methods, e.g. "Private Aggregation of Teacher Ensembles" (PATE) [15]. The federated learning architecture proposed by Papernot et al. [15] yields scalable and private distributed learning. However *PATE* is not topic to this thesis.

2.3 Membership Inference

Membership Inference attacks (MIA) reason about the existence of a given data point in the training set of a ML model. Generally, this assumes an *honest-but-curious attacker*. The attack formulates that there is a target model under attack, which is trained with dataset $D_{train}^{target} \sim D$ and validated

using $D_{test}^{target} \sim D$ with the restriction $D_{train}^{target} \cap D_{test}^{target} = \emptyset$. Given x as input into a MIA model, it should either predict "*in*" if it is part of D_{train}^{target} or "*out*" for not being part of training set. The adversary does not take any chance to change or exploit the target model, therefore MIA is considered a passive attack.

Definition 2.3.1. Membership experiment $Exp^M(\mathcal{A}, A, n, \mathcal{D})$
Let $\mathcal{A}$ be an adversary, A be a learning algorithm, n be a positive integer, and $\mathcal{D}$ be a distribution over data points (x, y). The membership experiment proceeds as follows:

1. Sample $S \sim \mathcal{D}_{test}^{target}$ and let $A_S = A(S)$
2. Choose $b \leftarrow \{0, 1\}$ uniformly at random.
3. Draw $z \sim S$ if $b = 0$ or $z \sim D_{train}^{target}$ if $b = 1$
4. $Exp^M(\mathcal{A}, A, n, \mathcal{D})$ is 1 if $\mathcal{A}(z, A_S, n, \mathcal{D}) = b$ and 0 otherwise. $\mathcal{A}$ must output either 0 or 1.

The Definition 2.3.1 is modified and taken from Yeom et al. [16] and specifies a formal *membership inference experiment.*

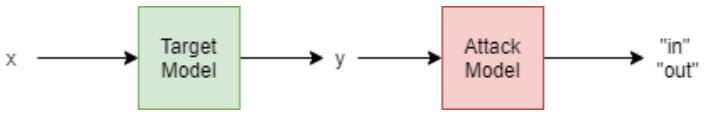


Figure 2.3: Membership Inference attack setting.

In Figure 2.3 a visualisation of the membership inference attack can be seen. The outputs from the target model are used to train the attack model. The success is strongly depending on the overfitting of the target model.
In related literature there are some different approaches to infer information about the membership of individuals in the training set of a target model. The first, the **black-box model**, does not take any internal features of the target model into account. Instead so-called "Shadow Models" are used, which try to mimic the target models output with respect to their inputs [2]. Since black-box model only exploit external features of the target model, the shadow model can only learn the output labels $f(x)$ with respect to input x. The realistic use-case of this setting is, e.g. cloud providers that use auto-ml solutions for their customers. Whereby the customer never comes to see the actual ML model.
Leino et al. [17] suggest a **naive attack** to compare the black-box attack.

Just like the black-box attack, the naive attack also merely requires arbitrary access to the target model. Given a data point x if the predicted outcome $f(x)$ is correct, assume x to be a member of the training set [17]. This attack is indeed naive, and achieves worse results than the black-box model [17].
As Leino et al. states, the greater the *generalization gap* is, measured e.g. by taking the difference from training and validation accuracy of the target model, the better the Membership Inference attack works. Despite the generalization gap, Leino et al. states, that the "idiosyncratic" use of features within the training of a model, reinforces the accuracy of MIA. When during training a certain data point is used, which is not in the representative population, particular features could be extracted and used to encode learned training samples. These features are called *evidence of membership*.
Another MIA-setting is the grey-box attack [18]. Truex et al. [18] defines this as an attack where the adversary has knowledge about the target model or training data. Furthermore the adversary has *specialized population-level knowledge*.

White-Box

In contrast to the black-box and naive attack model, the white-box model assumes the adversary to have access to the entire configuration of the target model, this includes the ANN architecture, parameters, and the data population. Nasr et al. compare different white-box approaches by using different components of the target model to train an attack model, these differ completely from the one suggested by Leino et al. They show, that training a MIA model with the gradients, layer outputs, labels and loss from the target model for every training and test input, they receive an attacker outperforming the black box variant [9].

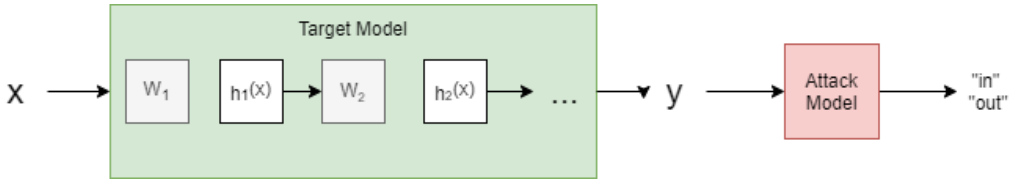


Figure 2.4: Membership White-Box Inference Attack setting.

In the MIA white-box setting more information about the target model can be leveraged to train a more efficient attack model. In particular by having access to the model $f(x, W)$ the adversary can not only observe the output

but also the gradients $\frac{\delta J}{\delta W}$, outputs of the hidden layer $h_i(x)$ and loss per sample. This can be seen as simplified in Figure 2.4. The white-box MIA approach from Nasr et al. especially the federated case is implemented and subject to this thesis for validating effective privacy from a target model. The White-Box Inference Attack is a realistic attack scenario in a Federated Learning network, because every participant shares the same model and hence has full knowledge about every model parameter.

2.4 Attribute Inference

Attribute Inference reasons about what an unknown attribute of a partially known individual might be. Instead of questioning if $x \in D_{train}^{target}$, the task here is to infer the value of the unknown and sensitive attribute i of $\vec{x}$. More concisely, the true projection $\pi_i(x)$ is searched. Yeom et al. [16] introduce an Attribute Inference attack that is similar to the naive Membership Inference attack: If f is the target model trained on D_{train}^{target} then if the loss $J(f(x;W),y)$ is low for a given x consider $x \in D_{train}^{target}$. For the Attribute Inference analogous: If x and x' differ in the same attribute and the loss of $J(f(x;W),y) < J(f(x';W))$ consider x to have the correct attribute value. The adversary of Yeom et al. simply searches over all possible attribute values for a certain attribute. In other words, given an individual x we want to infer a potentially sensitive target attribute $\pi_i(x)$ using all other partial information of x denoted as $\hat{\pi}(x)$.
An adversary could, for instance, be interested in the sexual orientation, gender or political view of a single individual in a publicly available dataset. Such datasets could be extracted from Facebook, Twitter or from other highly available and large online communities [19].

Definition 2.4.1. Attribute experiment $Exp^A(\mathcal{A}, A, n, \mathcal{D})$
Let $\mathcal{A}$ be an adversary, A be a learning algorithm, n be a positive integer, and $\mathcal{D}$ be a distribution over data points (x, y). The attribute experiment proceeds as follows:

1. Sample $S \sim \mathcal{D}_{train}^{target}$ and let $A_S = A(S)$
2. Draw $z \sim S$
3. $Exp^M(\mathcal{A}, A, n, \mathcal{D})$ is 1 if $\mathcal{A}(\hat{\pi}(z), A_S, n, \mathcal{D}) = \pi(z)$ and 0 otherwise.

The Definition 2.4.1 shows a formally defined *attribute inference experiment*.

It is taken from Yeom et al. [16] and was modified to fit the notation.

2.5 Differential Privacy

Differential Privacy (DP) constitutes a strong definition for anonymization techniques [1]. The notion DP is formally defined by Dwork et al. [7]. The goal of DP is to limit the information disclosure of *statistical databases*. Statistical databases only allow queries with aggregated results, like summing or averaging over multiple entries.
Using DP a data scientist should be capable of learning information about the population but still ensuring privacy for individual entries in the database. More specifically, a data scientist can not be sure about whether an individual is part of a database or not, by querying the database.
DP is enforced by adding noise to increase ambiguity about the underlying dataset. Thus, the success of inference attacks like *Membership Inference* is lowered. If an adversary can not even tell if an individual is part of a dataset, he is not able to infer any other information about individuals, hence increasing privacy for a particular application.

ϵ-DP

Dwork et al. [7] introduced the notion of ϵ-Differential Privacy (ϵ-DP) in 2006. This mathematical concept states a parameter for which queries to a statistical database differ with a low probability for a low ϵ if they are neighboring. A statistical database D' is neighboring to a database D if they differ in at most one individual. Statistical databases describe data which can only be queried by a set of operations or functions $f(D)$ using aggregation properties like a *sum* or *average* of multiple individuals [20]. In the following, ϵ-DP is formally defined.

Definition 2.5.1. ϵ-DP
An algorithm $\mathcal{M}$ satisfies ϵ-DP, where $\epsilon \geq 0$, if and only if for any datasets D and D' that differ on *one element*, we have

$$\forall t \in Range(\mathcal{M}) : Pr[\mathcal{M}(D) = t] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') = t]$$

Taken from Li et al. [20].

As seen in Definition 2.5.1, for all possible outputs of a *randomized* algorithm, a.k.a. mechanism, $\mathcal{M}$ on two neighboring databases, the probability that both outputs result in the same value t should be smaller than $\exp(\epsilon)$

to enforce DP for a certain ϵ [20, p. 7]. $\epsilon = 0$ is said to provide perfect privacy, because the output of a mechanism $\mathcal{M}$ is not depending on a database anymore. This however, would render its function useless, because it would give no information at all, about the underlying population. Therefore ϵ-DP provides privacy for sacrificing utility.
Definition 2.5.1 can be restated as:

$$\forall t \in Range(\mathcal{M}) : \frac{Pr[\mathcal{M}(D) = t]}{Pr[\mathcal{M}(D') = t]} \leq \exp(\epsilon)$$

with $\frac{0}{0} = 1$.
A small ϵ contributes to a more similar probability distribution for both datasets.
To understand the intuition of DP, Li et al. provide an "opting-out"-analogy [20]: If a data scientist wants to publish $\mathcal{M}(D)$ but a single individual has privacy concerns and is not consenting, the easiest solution would be to remove the individual's entry from D yielding D' with $\mathcal{M}(D')$. However receiving privacy protection by removing individuals is infeasible, since protecting everyone's data means removing everyone's entry (for example a row in a relational database). ϵ-DP can be seen as approximation of "opting-out", since the contribution of an individual entry to $\mathcal{M}(D)$ is kept small. This is achieved by ensuring that for any output t, it is similar likely one will see the same output even if a single individual (t) is removed.
DP is applied by using a mechanism $\mathcal{M}$ that enforces the Differential Privacy property by perturbing the result of $f(D)$. Dwork et al. defines the notion of *Privacy Loss* in following definition.

Definition 2.5.2. Privacy Loss
For two neighboring databases D, D' and an output $t \in Range(M)$,

$$\mathcal{L} = \ln\left(\frac{PR[\mathcal{M}(D) = t]}{PR[\mathcal{M}(D') = t]}\right)$$

Definition 2.5.2 shows the equation for the privacy loss. If output t is more likely for D than D' $\mathcal{L} > 0$ and vice versa.

Approximate DP

(ϵ, δ)-DP is a generalization of ϵ-DP and sometimes called *Approximate Differential Privacy*. The upper bound of ϵ-DP is allowed to be violated to some degree controlled by parameter δ. More concretely, ϵ-DP holds for $1-\delta$ probability. $(\epsilon, \delta = 0)$-DP equals ϵ-DP [7]. In practice δ should be smaller than $\frac{1}{n}$ with n being the number of individuals in the database.

Definition 2.5.3. (ϵ, δ)-DP
A mechanism $\mathcal{M}$ satisfies (ϵ, δ)-DP, where $\epsilon \geq 0$ and $\delta = [0, 1]$, if and only if for any datasets D and D' that differ on *one element*, we have

$$\forall t \in Range(\mathcal{M}) : Pr[\mathcal{M}(D) = t] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') = t] + \delta$$

As seen in Definition 2.5.3, δ is additively applied to the multiplicative from Definition 2.5.1. (ϵ, δ)-DP allows to find an output t that is much more likely to occur in D' than in D. The absolute privacy loss $|\mathcal{L}|$ is bounded by ϵ with probability $1 - \delta$.

Laplace mechanism

A frequently used ϵ-DP mechanism is the Laplace mechanism. This mechanism is suited for functions $f(D)$ that return scalar values. Assuming a dataset of lung cancer patients, $f(D)$ could, for example, return the average amount of cigarettes per day, a patient smoked in the last 15 years [20]. The definition by Dwork et al. is seen in Definition 2.5.4 [7].

Definition 2.5.4. Laplace mechanism
Given any function f, the Laplace mechanism is defined as:

$$\mathcal{M}_L(D, f(\cdot), \epsilon) = f(D) + (\mathcal{X}_1, \ldots, \mathcal{X}_k)$$

where $\mathcal{X}_i$ are i.i.d. random variables from the Laplace distribution.

To ensure $f(D)$ to be differentially private, the outcome has to be perturbed. This is done by adding a random variable $\mathcal{X}$:

$$\hat{f}(D) = f(D) + \mathcal{X}$$

Where $\mathcal{X}$ is a random variable drawn from a Laplace distribution. Despite needing $\hat{f}(D)$ to be an unbiased estimate of $f(D)$, following condition must hold:

$$\forall t, \frac{PR[\hat{f}(D) = t]}{PR[\hat{f}(D') = t]} = \frac{PR[f(D) + \mathcal{X} = t]}{PR[f(D') + \mathcal{X}' = t]} = \frac{PR[\mathcal{X} = t - f(D)]}{PR[\mathcal{X}' = t - f(D')]}$$

where both $\mathcal{X}$ and $\mathcal{X}'$ are drawn from the Laplace distribution. Let $d = f(D) - f(D')$ following must hold:

$$\forall x, \frac{PR[\mathcal{X} = x]}{Pr[\mathcal{X}' = x + d]} \leq \exp(\epsilon) \tag{2.3}$$

If Equation 2.3 holds for every possible d, then $\hat{f}(D)$ is ϵ deferentially private. Instead evaluating every possible d, the **maximum** difference is used, i.e. the **global sensitivity**. As formally stated in Definition 2.5.5, the global sensitivity measures the maximum impact an individual can have on the outcome of a function f [20].

Definition 2.5.5. Global Sensitivity ($\ell 1$)
Let $D \simeq D'$ denote that D and D' are neighboring. The global sensitivity of a function f, denoted by Δf, is given below

$$\Delta f = \max_{D \simeq D'} |f(D) - f(D')|$$

This concept is needed in order to ensure that Definition 2.3 holds for all neighboring D and D'. Furthermore the probability density function should have the property, that moving no more than Δf on the x-axis it should increase or decrease the probability by no more than $exp(\epsilon)$ [20]. The largest density should lie on $f(D)$ or $f(D')$ respectively.

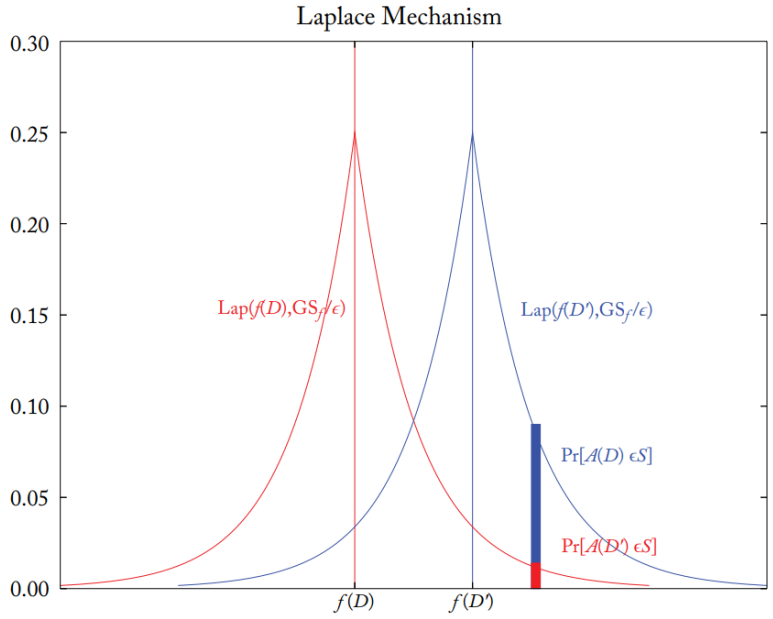


Figure 2.5: Laplace distributions for two neighboring datasets around Δf. Taken from Li et al. [3, p. 13].

In Figure 2.5 two probability density functions are seen both for $f(D)$ and $f(D')$ with noise sampled from a Laplace distribution. The difference of

$f(D)$ and $f(D')$ on the x-axis is Δf, i.e. the global sensitivity. The higher the gap between the peak of both distributions, the lower the privacy and consequently the more noise needs to be added.
The Laplace mechanism samples noise from the Laplace distribution:

$$Lap(x|\mu, b) = \frac{1}{2b} \exp(\frac{-|x-\mu|}{b})$$

The parameters are fixed to $\mu = 0$ and scale parameter b to $b = \frac{\Delta f}{\epsilon}$. This leads to:

$$\mathcal{X} \sim Lap(x|0, \frac{\Delta f}{\epsilon})$$

In the following, an example proof for Differential Privacy for the Laplace mechanisms is illustrated, to allow the interested reader to further understand DP.
Taken from Li et al., Proof 2.5 shows that $\frac{1}{2b} \exp(\frac{-|x-\mu|}{b})$ yields ϵ-DP [20].

Proof. For any function f, the Laplace mechanism $\mathcal{M}(d) = f(D) + Lap(0, \frac{\Delta f}{\epsilon})$ satisfies ϵ-DP.

$$\begin{aligned}
\frac{Pr[f(D) + \mathcal{X} = t]}{Pr[f(D') + \mathcal{X} = t]} &= \frac{Pr[\mathcal{X} = t - f(D)]}{Pr[\mathcal{X} = t - f(D')]} = \frac{Pr[Lap(0, \frac{\Delta f}{\epsilon}) = t - f(D)]}{Pr[Lap(0, \frac{\Delta f}{\epsilon}) = t - f(D')]} \\
&= \frac{\frac{1}{2b} \exp(\frac{-|t-f(D)|}{b})}{\frac{1}{2b} \exp(\frac{-|t-f(D')|}{b})} = \exp\left(\frac{|t - f(D')| - |t - f(D)|}{b}\right) \\
&\leq \exp\left(\frac{f(D) - f(D')}{b}\right) \leq \exp\left(\frac{\Delta f}{b}\right) \leq \exp(\epsilon)
\end{aligned} \tag{2.4}$$

Therefore:

$$b = \frac{\Delta f}{\epsilon}$$

□

Proof 2.5 is slightly modified and taken from Li et al. [20, p. 14] to show that sampling noise $\mathcal{X} \sim Lap(0, \frac{\Delta f}{\epsilon})$ yields ϵ-DP for a given global sensitivity.

Gaussian mechanism

The second mechanism that will be used within this thesis, is the *Gaussian mechanism*. The Gaussian mechanism, compared to the formerly described Laplace mechanism, samples noise from a Gaussian distribution, hence the

name. While the Laplace mechanism achieves $(\epsilon, 0)$-DP, the Gaussian mechanism yields (ϵ, δ)-DP. Due to the potentially high dimensionality the Gaussian mechanism uses the Euclidean norm to calculate the global sensitivity, i.e. $\ell 2$-sensitivity, seen in Definition 2.5.6.

Definition 2.5.6. Global Sensitivity ($\ell 2$)
Let $D \simeq D'$ denote that D and D' are neighboring. The global sensitivity of a function f, denoted by Δf, is given below

$$\Delta_2 f = \max_{D \simeq D'} ||f(D) - f(D')||_2$$

Recall that the global sensitivity indicates how much noise needs to be added to achieve (ϵ, δ)-DP.
Let f be a d-dimensional function. The Gaussian mechanism adds noise from a Gauss distribution with parameter $\mu = 0$ and a certain variance σ^2 yielding $\mathcal{N}(0, \sigma^2)$. The Gauss distribution is defined by:

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2)$$

Dwork et al. proved that if the scale of noise satisfies:

$$\sigma \geq \sqrt{2\log(\frac{1.25}{\delta})}\frac{\Delta_2 f}{\epsilon}$$

the Gaussian mechanism guarantees (ϵ, δ)-DP, however the proof is omitted in this thesis, due to its length [7]. The Gaussian mechanism is generally a popular choice, since its relaxation yielding (ϵ, δ)-DP. Besides, the Gaussian distribution is a closed form (e.g. closed under addition). Data scientists can therefor more easily analyze Differential Privacy under Composition.

Composition

The formerly discussed mechanisms provide Differential Privacy for a single function evaluation (i.e. query). The common use case, however, is to answer a query several times or to combine multiple sub-queries to a more complex query. As Mcsherry points out, an adversary is capable to reconstruct the original value, when having unlimited query access [21]. Imagine having a Laplace mechanism applied to a function which returns the amount of lung cancer patients of a dataset. Due to the Law-of-large-numbers and the symmetric noise distribution of above mechanisms, querying and averaging this information after a huge amount of times, eventually, the true value can

be obtained.
Dwork et al. addresses this issue with the *Sequential Composition Theorem* [7]. The idea is to add up (ϵ, δ) used by every call to a query or sub-query.

Definition 2.5.7. General Sequential Composition (ϵ, δ)-DP
Let $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_k$ be k mechanisms
that satisfy (ϵ_1, δ_1)-DP, (ϵ_2, δ_2)-DP,...,(ϵ_k, δ_k)-DP, respectively, with respect to the input dataset. Publishing

$$t = (t_1, t_2, ..., t_k)$$

where

$$t_1 = \mathcal{M}_1(D), t_2 = \mathcal{M}_2(t_1, D), ..., t_k = \mathcal{M}_k((t_1, ...t_k), D)$$

satisfies $(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i)$-DP.

Definition 2.5.7 (taken from Li et al. [20]) show that the privacy parameters ϵ and δ of k mechanisms sum up. Vice versa the summation can also be interpreted as having a budget that needs to be divided under sequential composition and consumed by individual steps in an algorithm [20, p. 9]. For this reason ϵ is often referred to as *privacy budget*. In the homogeneous case, this yields $(k\epsilon, k\delta)$-DP, which would imply linear degradation of privacy with the number of mechanisms k in the composition. However, Dwork et al. proofed that for $\delta > 0$, this bound is not tight and advanced composition methods can achieve higher privacy guarantees [7]. There a multitude of techniques available, to effectively track ϵ under compositional queries, e.g. *Moments Accountant* [1], *Strong Composition Theorem* [7] or the *Advanced Composition Theorem* [22].
Mironov et al. introduce *Rényi Differential Privacy* which provides a tighter bound for mechanisms under composition [23]. This is due to the fact, that the underlying assumption from the General Sequential Composition is, that every time an attacker queries $\mathcal{M}$ he becomes more and more certain whether the result is from D. Realistically however, the attacker could be more biased towards D' as well. RDP is based upon the Renyi entropy, which is a generalization to e.g. the Shannon entropy, Hartley entropy or min-entropy. In the information theory, entropies are used to measure the amount of information for a statistical random variable. A divergence, however, measures the difference between two statistical random variables. Renyi divergence, in that case, uses Renyi entropy to calculate the difference between two statistical systems, as seen in Definition 2.5.8.

Definition 2.5.8. Renyi-Divergence
Let $\alpha > 1$ and $p \sim P$ and $q \sim Q$ be two random variables drawn from two

probability distributions, Renyi-Divergence is defined as:

$$D_\alpha(P||Q) = \frac{1}{1-\alpha} log\left(\sum_{i=0}^{n} \frac{p_i^\alpha}{q_i^{\alpha-1}}\right)$$

The definition is taken from Mironov et al. [23], although modified to fit to this thesis' notation.

It is proven, that for $\lim_{\alpha\to 1}$, $D_1(P||Q)$ is equivalent to the Kullback-Leibler-Divergence.
The motivation to use RDP lies in the obvious fact that a mechanism $\mathcal{M}$ is ϵ-DP if and only if its distribution over any two neighboring datasets satisfies: $D_\infty(\mathcal{M}(D)||\mathcal{M}(D')) \leq \epsilon$ for $\lim_{\alpha\to\infty}$ [23].
The sequential composition theorem for RDP is defined as: $(\alpha, \sum_{i=0}^{k} \epsilon_1)$-DP.

2.6 Differentially Private Learning

Differential Privacy is enforceable on each record $d \in D$, commonly referred to as *Local Differential Privacy* (LDP), or on an aggregate function $f(D)$, referred to as *Central Differential Privacy* (CDP). In this thesis both approaches were validated in the experiments and thus a brief introduction is necessarily provided in the next following sections.

Central Differential Privacy

Central Differential Privacy (CDP) is used to perturb a aggregation function $f(\cdot)$. A CDP mechanism has to fulfill Definition 2.6.1.

Definition 2.6.1. (ϵ, δ)-Central Differential Privacy
A mechanism $\mathcal{M}$ is said to be (ϵ, δ)-central differentially private if D and D' differ in at most one element and all outputs $\forall t \in Range(\mathcal{M})$.

$$PR[\mathcal{M}(D) = t] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(D') = t] + \delta$$

The definition was taken from Bernau et al. [5]. In CDP, noise is not added to records in D_{train}^{target} but to an aggregation function during the training process. This is done by utilizing *differentially private gradient descent optimizers*, e.g. DPSGD or DPAdam [1]. The algorithms are modifications of the update rule seen in Section **Gradient Descent** 2.1. Particularly, they add noise to gradient updates during the backpropagation process while training an ANN. Counter-intuitively Abadi et al. claim that adding noise to the parameters after the training, renders the model unusable, thus the gradients have to be

perturbed during the training [1].
A differentially private optimizer represents a DP mechanism [5]. This DP mechanism $\mathcal{M}_{nn}$ updates the weights of the ANN in each training step t with $W^t = W^{t-1} - \eta(\tilde{g})$, where $\tilde{g} = \mathcal{M}_{nn}(\frac{\delta J}{\delta W^{t-1}})$ denotes the gradient. The gradients are perturbed by Gaussian noise and hyperparameter α is some scaling function on $\tilde{(}g)$. After a certain number of gradient updates, the mechanism $\mathcal{M}_{nn}$ yields a differentially private weight matrix W for the ANN. Because the amount of noise added by a mechanism depends on the global sensitivity, the gradients have to be *clipped*, in order to bound the sensitivity in the backpropagation process.
Hyperparameter $\mathcal{C}$, also called *Clipping Norm*, determines at what value the gradients are clipped. Additionally hyperparameter z, also called *noise multiplier*, controls the amount of noise that is sampled from the distribution and added to the clipped gradients. The noise distribution is Gaussian and therefore $\sigma = z \cdot \mathcal{C}$.

Local Differential Privacy

Sampling and adding noise to each entry $d \in D$ is called *Local Differential Privacy*. While CDP can be used by a trusted server, LDP is the preferred choice, if a central server can not be trusted. In this setting, the data is perturbed before the model is trained. To achieve LDP, noise is added to the data by $\mathcal{LR}$, as can be seen in Definition 2.6.2 [5].

Definition 2.6.2. Local Differential Privacy
A *Local Randomizer* $\mathcal{LR}$ is ϵ-local differentially private, if $\epsilon \geq 0$ and for all possible inputs v,v' and all possible outcomes $t \in Range(\mathcal{LR})$:

$$PR[\mathcal{LR}(v) = t] \leq \exp(\epsilon) \cdot Pr[\mathcal{LR}(v') = t]$$

The LDP experiments within this work are performed by using a local randomizer to independently perturb each entity $d \in \mathcal{D}$. However, since features can be highly correlated, the local randomizer has to repeatedly perturb the data. A local algorithm repeatedly invokes the $\mathcal{LR}$, and the resulting privacy guarantees are formally defined in Definition 2.6.3. Due to the sequential application of the local randomizer, privacy gradually degrades.

Definition 2.6.3. Local algorithm
An algorithm is ϵ-local, if it accesses the database $\mathcal{D}$ via $\mathcal{LR}$ within the following restriction: for all $i \in \{1, \ldots, |\mathcal{D}|\}$, if $\mathcal{LR}_i(i), \ldots, \mathcal{LR}_k(i)$ are the algorithm invocations of $\mathcal{LR}$ on index i, where each $\mathcal{LR}_j$ is an ϵ_j-local randomizer, for $1 \leq j \leq k$, then $\epsilon_1 + \ldots + \epsilon_k \leq \epsilon$.

Definition 2.6.2 is taken from Bernau et al. [5] who themselves based their definition on Kasiviswanathan et al. [24]. As seen in the formal definition, a $\mathcal{LR}$ perturbs data entries independently. From Definition 2.6.3 follows, that a local ϵ_i is a slightly weaker privacy guarantee than the global ϵ achieved from e.g. a CDP mechanism.

Randomized Response

Randomized Response is a ϵ-LDP mechanism which can be used for LDP as Local Randomizer [7, p. 30]. This technique was originally developed for the social sciences to evaluate illegal or embarrassing behaviors in surveys and hence to overcome *Non-response Bias*. To illustrate randomized response, imagine an embarrassing activity "P" and the query "Have you engaged in P in the past week?". But the n participants from the survey are instructed to answer "yes" or "no" about property P according to following schema, which also can be seen in Figure 2.6:

1. Flip a coin.
2. If **tails**, then respond truthfully
3. If **heads**, then flip a second coin and respond:
   (a) "Yes" if heads.
   (b) "No" if tails.

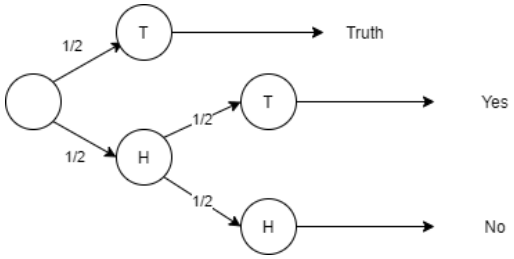


Figure 2.6: Randomized Response schema

The intuition behind randomized response and all other DP mechanisms is that it provides *plausible deniability* [7]. Every participant can deny the outcome, due to the random coin flips.

Proof. Randomized response is $ln(3)$-differentially private for two possible events using a coin flip.

$$\frac{Pr[Response = Yes|Truth = Yes]}{Pr[Response = Yes|Truth = No]} = \frac{Pr[Response = No|Truth = No]}{Pr[Response = No|Truth = Yes]} = \frac{1/2 + 1/4}{1/4} = 3 \tag{2.5}$$

Inserting this into Definition 2.5.1 yields $\epsilon_i = ln(3)$. □

Although every single response, i.e. entry $d \in D$, will be perturbed, the expected percentage of individuals in the population who have property P can be approximated by:

$$Pr[Response = Yes] = \frac{1}{4} + \frac{1}{2} \cdot \hat{p} \Rightarrow \hat{p} = (Pr[Response = Yes] - \frac{1}{4}) \cdot 2$$

2.7 Related Work

This work aims to find suiting parameter ϵ for ANNs, which were trained in a federated fashion. DP is enforced using CDP and LDP mechanisms and evaluated using MI and AI attacks against the confidentiality of these models. Hence this work is related to research fields in these three components: White-Box MIA, Federated Learning and DP.

White-Box MIA

Nasr et al. evaluated white-box Membership Inference attacks against both, centrally and distributed (federated) learned models [9]. The FL protocol used in their work is specified by Konečny et al. [25]. They show that for a small number of participating clients it is possible for an adversary to infer sensitive information in a global Membership Inference Attack as well as for an local attacker. Their attack mode learns directly from the target models internal parameters. Furthermore they assume the adversary to possess a fraction of the Data Owners sensitive data. These stronger assumptions on the adversary increase the overall strength of the MI attack in contrast to black box MI.
Hitaj et al. proposes an attack model using a *Generative Adversarial Network* GAN to infer sensitive data in a collaboratively learned model [26]. They stat, that their GAN attack model is able to archive high accuracy

even when DP is applied. Nasr et al. however point out, that their FL protocol is not standardized. More specifically, they update the global model each time the clients trained on a single *mini-batch*. Furthermore, their adversary constitutes an active attack. An local attacker injects generated samples from the GAN and labels them according to another class. This way, the other clients need to invest more work to distinguish between these classes. Eventually evidence of membership will be encoded into the mislabeled class, making it easier to infer.

FL

The currently active research efforts in FL focus training with *heterogeneous* statistical distributions within the datasets of the clients. Algorithms like Matched Averaging (FedMA) [27], or FedProx [28] try to tackle this problem. FedMA applies a *layer-wise*, instead like in FedAVG a *coordinate-wise* average. FedProx modifies the federated SGD slightly by having a proximity term added. These research topics are not concerning privacy, whatsoever. Although for different federated algorithms, probably other DP-parameters have to be evaluated. One differentially private technique to fit a ML model distributively is *PATE* [15]. Given a number of Data Owner with disjoint sensitive data. PATE refers to each client as individual *Teacher*. The so called *aggregate teacher* takes for a given input data the prediction votes from the teacher ensemble. The aggregation then, adds Laplacian noise to the vote histogram. DP is now enforced due to the fact, that if multiple teacher agrees on the same label it is not depending on a single model. Also DP is enforced, because the aggregate teacher yields the prediction with the highest noisy vote from the ensemble. This mechanism is called *max-of-Laplacian* mechanism. The last step in PATEs training involves a student model. The student model uses unlabeled and non-sensitive data and combines them with a prediction label by the aggregate teacher. An adversary should have no access to the aggregate teacher or the teacher ensemble at all. Only the student model should be publicly available.

DP

Abadi et al. present a way to apply a Gaussian mechanism to the Gradient Descent Optimizer for ANNs [29]. Bernau et al. compare this CDP mechanism and LDP mechanisms for different ML models. For comparability these datasets are also used in this thesis. For the distributed case of training a ML model Agarwal et al. present a way to apply a central differentially private Binomial mechanisms to Federated Learning [14]. Although their work in-

tends to find a secure and communication efficient trade-off while this work's goal is to propose a good utility-privacy-trade-off. This thesis however, follows the same approach as the one from Wu et al. [30]. Wu et al. present a way to apply DP to collaborative learning. Their protocol consists of Data Owners and a aggregator (referred to as *learner*). The Data Owners train one epoch on their local model with their sensitive data. Afterwards the aggregator queries the Data Owners and receives their perturbed gradients. They prove that the quality of trained ML models scales inversely with the squared privacy budget and the squared size of the dataset. This metric will also be validated in this thesis. However, they enforce DP only through CDP and are not evaluating the effective privacy by using Inference Attacks [30].

2.8 Notation

The purpose of this last section is to provide Table 2.1 of the notation used throughout this thesis. The notation mostly matches with Bernau et al. [5].

Notation	Description
D	Dataset.
$\mathcal{DO}$	Set representing Data Owners 1...n, i.e. $\mathcal{DO} = \{\mathcal{DO}_1, \ldots, \mathcal{DO}_n\}$.
$D^{train}_{\mathcal{DO}_k}$	Training Data for local model of $\mathcal{DO}_k$
$D^{test}_{\mathcal{DO}_k}$	Test Data for local model of $\mathcal{DO}_k$
$\mathcal{A}$	Adversarial aggregator
$D^{train}_{target_k}$	Training Data for the target model from data owner k.
$D^{test}_{target_k}$	Test Data for the target model from data owner k.
$\mathcal{AM}$	Attack Model i.e. the model used to infer membership
$D^{in}_{target_k}$	Data from data owner k labeled *in*.
$D^{out}_{target_k}$	Data from data owner k labeled *out*.
D^{train}_{attack}	Train Data for the attack model.
D^{test}_{attack}	Test Data for the attack model.
$\mathcal{GM}$	Global Model obtained from the aggregation of the local models.
$\mathcal{TM}$	Target Model i.e. the model used to train the attack model.
x	A specific data point $\vec{x}$ for convenience reasons the vector arrow is dropped.
$J(f(x; W), y)$	The loss function w.r.t. input x and true value y.
$f(x)$	A concrete function e.g. database query.
$h_i(x)$	output of i-th hidden layer.
$\frac{\delta J}{\delta W_i}$	Weight gradients of i-th layer.
$X = x_0, \ldots, x_n$	Certain datapoints.
$Y = y_0, \ldots, y_n$	Corresponding labels for X.
η	Learning rate parameter.
$\pi(x)$	A certain projection of data point x.
$\pi_i(x)$	The ith attribute of data point x.
ϵ	Differential Privacy parameter.
$\mathcal{M}$	ϵ-DP mechanism.

Table 2.1: Notation used in this thesis.

Methodology

Within this chapter, first the research hypothesis is stated and afterwards the threat model under which the research hypothesis is validated, is being discussed. The following implementation section describes in detail how the experiments are performed. The chapter concludes with a definition of evaluation metrics for comparing white-box MI attacks.

3.1 Hypothesis

Nasr et al. [9] suggest a white-box Membership Inference Model and performed them in numerous experiments. They have shown, that MI attacks are successful against Federated Learning environments. Using this knowledge, this work hypothesizes, that MI attacks are suited for measuring the effective privacy enhancement in the FL context. If the hypothesis is correct, there should be a trade-off sweet-spot, which could act as a general guideline for data scientists in the future.

3.2 Threat Model

All experiments within this thesis make the same assumptions about the MI adversary. Thus, every experiment within the thesis considers the same threat model, which is introduced in the following.

Type of Adversary

For all experiments, it is assumed that the adversary is not actively changing the model's behavior, but instead merely observes input, output, and internal

parameters of the model. In the field of cryptography research, this passive adversarial behavior is usually referred to as passive or *honest-but-curious* [31]. Compared to Nasr et al. this work will not consider the active attack strategies like exploiting the SGD or completely separating each data owner from the global update.

Adversarial Knowledge

The adversary is assumed to have full access to the trained ML model, i.e., the weights and losses in addition to the input and output. To perform the attacks, the adversary also knows a certain portion of the target model's training and test data. This approach is very similar to Nasr et al. [9]. In the Federated Learning process, there are two potential adversaries. In both scenarios the attacker tries to infer potentially sensitive information about individuals in the training set of the target model from a data owner. Either with membership or attribute inference.

Global Attack The first attack scenario is called *Global Attack*. Here, the adversary acts as the aggregator who tries to infer sensitive information about the individuals of the training set from the data owners. In the Global Attack, the aggregator isolates and stores each data owner's update before aggregating it. In consequence, T local models are stored by the aggregator per data owner.

Local Attack In the *Local Attack*, an adversarial data owner tries to infer sensitive information about other data owners participating in the training. The adversary stores T global models, for T non-consecutive epochs. Furthermore, the attacker has a certain fraction of the train- and test data of the data owner under attack.

Nasr et al. [9] report that the best results are achieved, when the attacker has access to models from T non-consecutive epochs. Furthermore, they state, that they tried with $T = 5$ because they had hardware limitations and could not test larger T, although they think, larger T yield better results. Lastly, the attribute inference attack additionally assumes some domain knowledge about the sensitive attributes.

3.3 Implementation

In this section all implementation details regarding the Federated Learning Framework, Attribute and Membership Inference Attack model are explained.

Federated Learning Framework

Federated Learning (FL) is used to train a ML model distributed across multiple data owners. There is a central instance that monitors the training while several participants actually execute the training. In FL the central instance assures that everyone complies with the protocol. The implementation in this work is based on the naive synchronized algorithm of Konečny et al. [25]:

1. A subset of connected clients is selected, each of which downloads the current model.
2. Each client in the subset computes an updated model based on their local data.
3. The model updates are sent from the selected clients to the sever.
4. The server aggregates these models to construct an improved global model.

The naive algorithm is sufficient, because the solely purpose of this framework is to run experiments on only a few simulated devices, which is typical for a b2b environment. Optimizations to decrease the communication rounds are not necessary since in this thesis the processes run on the same physical machine. This algorithm is best implemented using a client-server-architecture model. The FL protocol in this thesis might differ from Nasr et al. research paper, since they did not clarify which protocol they used for their experiments. Although, the protocol in this work is said to be state-of-the-art [32].

Components

The software architecture follows the implementation suggestion of Bonawitz et al. [32], which is based on an actor model, i.e. the framework uses a client-server-architecture, where the components communicate over message passing. There are two main components, as seen in the UML component diagram in Figure 3.1: The server (aggregator) and the clients (data owners).

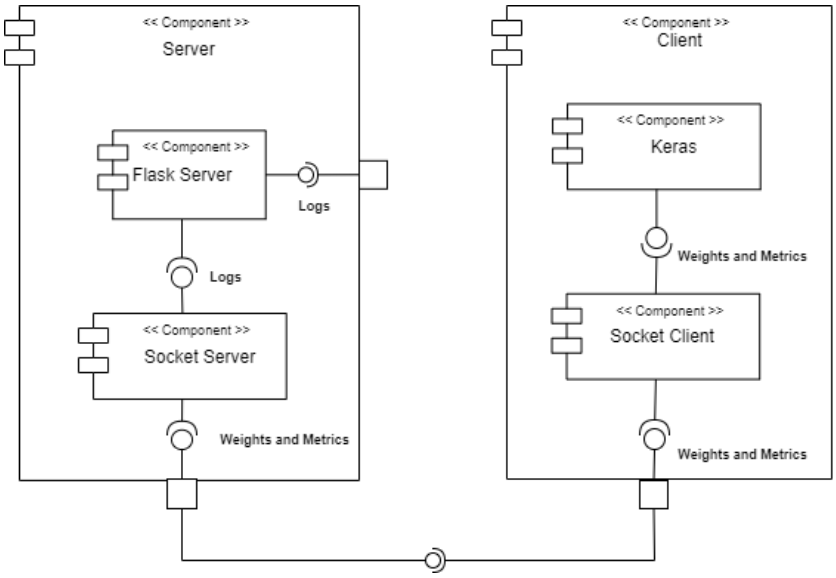


Figure 3.1: UML Component Diagram of the Federated Learning Framework.

The server listens on a multi-threaded TCP Socket (websockets) assuring a bidirectional real-time communication. A HTTP-Server (Flask) is used to serve training logs and metrics via a web GUI. Each thread handles a different client. After a client connects the first time, the server sends the serialized ML model template. The clients, each consisting of a single threaded websocket client, receive the model, execute batch gradient descent for a specific amount of epochs and eventually send the updated local models weight matrix back to the server. This procedure corresponds to the formal Algorithm 1 seen in the previous chapter.
In the following, a short outline of the two main components is presented:

Client A client holds a reference to a local model and keeps an unique identifier retrieved from the server. When the training is finished, the clients save their local models instead of just the weights. This way, the optimizer state is saved as well. A derived class called "AdversarialClient" denotes a specific client, who saves the received global model every T epochs before it starts its local training.

Server The server holds a reference to a global model, a list of connected clients and the queue containing the latest client messages. The queues capacity is set to $max(C \cdot n, 1)$, i.e. the amount of clients. A derived class called "AdversarialServer" denotes a specific server, who saves the isolated local updates from each client for T epochs.

All clients, as well as the server, run in their own respective **isolated** process. This is to enforce the memory-safe paradigm: "Do not communicate by sharing memory; instead, share memory by communicating." [33].

Communication

Based on the actor model, in the client-server-dialog, each message is associated with a particular event. The simplicity lies in the few events that are needed:

1. on *connect*
   The client connects successfully to the server. The server answers with the event *init* and the model template.
2. on *init*
   The client receives the training parameters and emits the event *ready*.
3. on *ready*
   The server enqueues the message and waits for $m <= n$ clients to be ready. Eventually it broadcasts the event *request_ update*. This results in a so called "barrier" which ensures synchronicity among the processes.
4. on *request update*:
   The client receives the current global model and the instruction to train E epochs on their local model. He answers with the *client_ update* event, their updated weights, their training data size, their training loss and accuracy as well as their test loss and test accuracy.
5. on *client update*:
   The server enqueues the message and waits for $max(C \cdot K, 1)$ clients to have sent their updated weights. Then it aggregates the weights and evaluates the global loss and accuracy on the validation data. Finally, it broadcasts *request_ update* again.

Dialog 3.3 repeats until the server monitors an insignificant change in the validation loss for a specific amount of epochs i.e. effectively enforcing "early stopping". The server is not using a global event loop to decide when to take action. Rather, a concurrency safe queue, is used, to store the client messages. The thread that handles the $C \cdot n$-th message, aggregates the metrics and weights. Then the queue is flushed and another instruction to train the local models is broadcast to the clients.
The messages and the "Keras" model are "JSON" serialized python dictionaries, while the weights are serialized and compressed using "Numpy". For this

thesis, different data encoding schemes were evaluated, for instance: "MessagePack" and "Pickle". However, the best runtime and size trade-off was achieved using JSON with "Gzip" HTTP compression.

MIA Model Architecture

Given a data point (x, y), the adversary's objective is to determine the membership in the training set $D^{target}_{train_k}$ of the target model $f(x; W_k)$ from data owner k. For convenience, in the following explanation of the MI model, the index k for data owner k is dropped.
The classification task of the MI Attack is performed by the attack model, which makes use of different features from the target model. This section explains this specific attack model architecture and how the attack model inputs relate to the target model. Furthermore, the generation of attack features and the training of both models are described. The attack model, implemented within this work, is based on the white-box MI attack proposed by Nasr et al. [9].

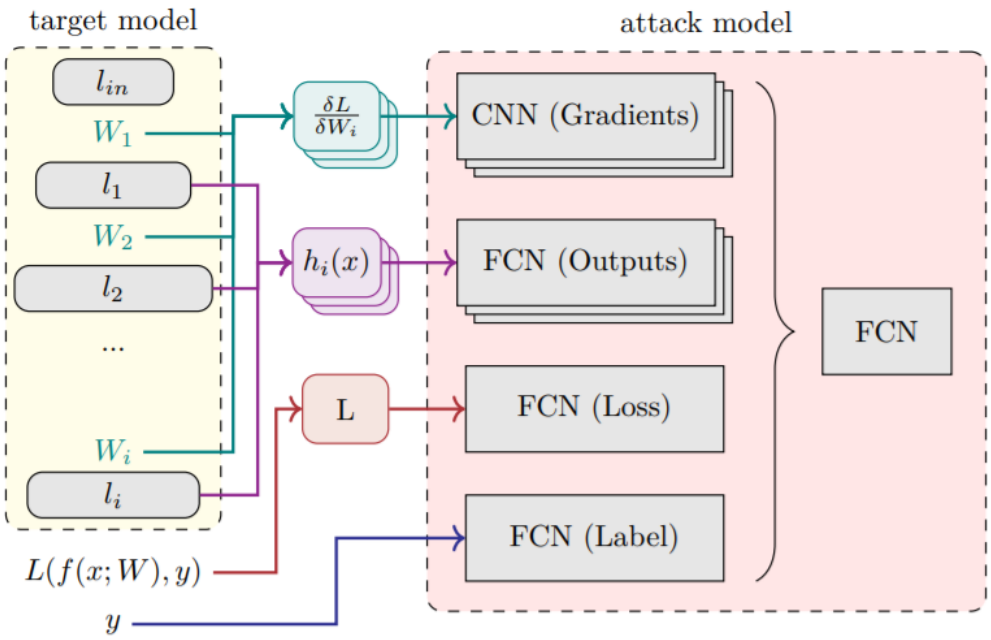


Figure 3.2: White-Box Membership Inference Attack Architecture [9].

The MI attack makes use of all internal and external features of the target model under attack, including: the loss $J(f(x; W), y)$, the hidden layer outputs $h_i(x)$ of each hidden layer i, the gradients $\frac{\delta J}{\delta W_i}$ and the Label y. As seen

in Fig. 3.2, every feature is the input for a particular *component*. The gradient matrix constitutes the input for a *Convolutional Neural Network* (CNN). The CNN has a kernel size of $(1, n_p)$, where n_p is the number of nodes in the previous layer *to capture the correlation of the gradients in each activation function* [9]. The hidden layer outputs $h_i(x)$, the loss $J(f(x; W), y)$ and the one-hot-encoded Label y are input to a *Fully Connected Network* (FCN). Furthermore, in the Federated Learning-scenario, the adversary observes the target model for multiple epochs. Since in the FL context, there is no single point in time where the trained model is being deployed, the trained model is rather used by the data owners until it gets replaced by a new global model. In practice, this process may never stop at all [4]. In this particular case, when capturing the target model for T epochs, each input component's output is stacked together. For instance, instead of having only one loss feature L, here, the loss features are composed of $L = \{L^1, L^2, \ldots, L^T\}$ [9]. If the target model has i layers (without the input layer) and was observed T epochs, then the attack model consists of $2 \cdot i + 1$ components. The output of each component is merged and used as input for a final FCN component also called *encoder*. Eventually the last FCN outputs a single sigmoid activation value to perform the binary classification task for *in* or *out*. Nasr et al. evaluated different combinations of the components. Furthermore they extract information from every layer in the target model. In our experiments, the last two layers of the target model, with every information, were sufficient. Adding more layers did not change the outcome but elongated the attack training process immensely.

Name	Layer	Details
Output Component	2 Fully Connected Layers	Sizes: 128, 64 Activation: LeakyReLu Dropout: 0.2
Label Component	2 Fully Connected Layers	Sizes: 128, 64 Activation: LeakyReLu Dropout: 0.2
Loss Component	2 Fully Connected Layers	Sizes: 128, 64 Activation: LeakyReLu Dropout: 0.2
Gradient Component	Convolutional Layer	Kernels: 1000 Kernel size: $\mathcal{C}$ x Next layer Stride: 1
	2 Fully Connected Layers	Sizes: 128, 64 Activation: LeakyReLu Dropout: 0.2
Encoder Component	4 Fully Connected Layer	Sizes: 64, 4, 1 Activation: LeakyReLu Dropout: 0.2

Table 3.1: Modified White-box MIA model based upon Nasr et al. [9].

The overall architecture details can be taken from Table 3.1. The architecture differs slightly from Nasr et al. [9]. The *ReLu* activation functions from the original approach were replaced with *LeakyReLU* functions. This is due to the fact, that during training, we observed the attack model to sometimes stagnate and predict every sample as being "in". Compared to ReLu functions, LeakyReLu allows small gradients even when the neuron is not active. This averts the "Dying ReLu" phenomenon and allows the network to correct itself. Furthermore, *batch normalization* layers were added to every input layer to not only normalize every input batch-wise but also to add a regularization effect. Normalization is important, since attack features from multiple different epochs are stacked together. Some parts of the attack features like gradients and loss can show different statistical properties over different epochs and hence need to be normalized. At last, the convolutional layer for the gradient component is modified. With k being the amount of gradients and $\mathcal{C} = 4$ a fixed hyperparameter for the next layer, the kernel is now $\mathcal{C} \cdot k$ shaped.

Target Model Training

The training of the model under attack is an important factor deciding whether an attacker can succeed or not. A strongly overfitting target model makes an MI attack significantly easier since the target model performs extremely well on the training data and badly on the test data. This is likely to show in all attack features which are extracted from the target model. In order not to bias the experiments, the target model is trained to achieve a high test accuracy and hence a small train-test-accuracy gap. This ensures the experiments are as close as possible to a real-world scenario.
The target model in this thesis is trained using Federated Learning: Multiple data owner collaboratively train one global model. However in this implementation, the aggregator receives a fraction of the data to be able to enforce *early stopping*, while the data owners perform a 50 : 50 train/test split. Other public available implementations use the averaged validation loss of the clients to measure the overall performance as seen in Equation 3.1. This way the loss from all k $\mathcal{DO}$s is aggregated using a weighted average w.r.t. their training set size.

$$GlobalLoss \leftarrow \sum_{k=0}^{K} \frac{n_k}{n} J(f(x; W_k), y) \tag{3.1}$$

During this work, the experiments were tested with both measurements, and the results do not vary significantly. The data distribution for each data owner can be seen in Fig. 3.3.

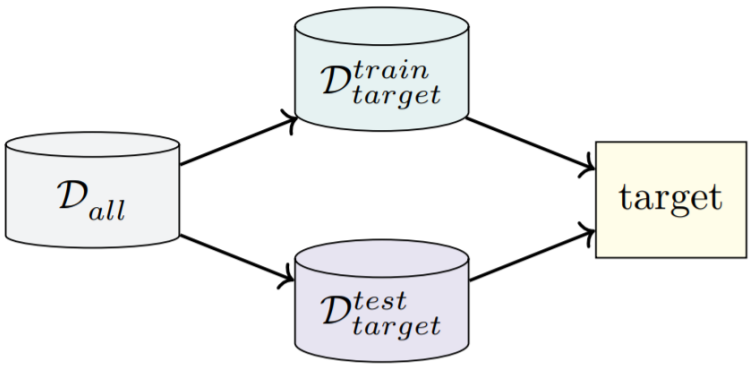


Figure 3.3: Dataset distribution for target model [5].

In this setting, we have $D_{target_k}^{train}, D_{target_k}^{test} \sim D$. D is split up 50 : 50, so that the attacker has the same amount of data not being present in the training set and data which was used during training. The target model is then

trained using $D^{train}_{target_k}$ for all data owner k. The aggregator decides when to stop training, by monitoring the validation accuracy. Nasr et al. used four clients to train the model [9]. More data owners lead to a performance loss on the attack model. Moreover, developers from SAP internally came to the conclusion, that $n = [4, 5]$ is commonly found in real word applications.

Attack Model Data Generation

To generate the data for the attack model training and evaluation, the T trained target models are loaded first. This means the same target model architecture, which is used during training, is used for attack feature extraction. The target models weights which were learned during training are loaded.

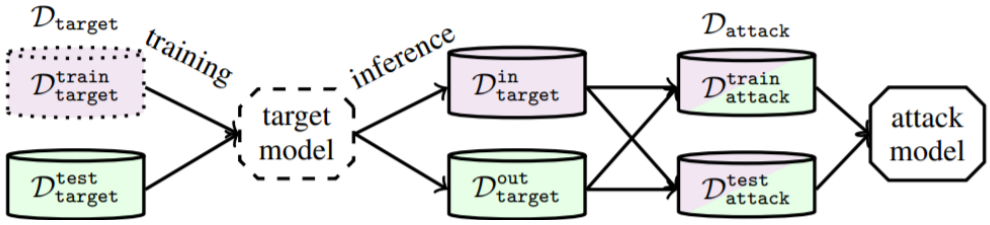


Figure 3.4: Dataset distribution for attack model [5].

To train the attack model, the first step is to load each data point from $D^{train}_{target_k}$ and $D^{test}_{target_k}$ from target data owner k. The attack features including the gradient matrix, hidden layer outputs, loss and label are extracted yielding $D^{in}_{target_k}$ for a data point with extracted features that were in the training set of the data owner k, and $D^{out}_{target_k}$ for the data with extracted attack features that were part of the target models test set. This procedure can be observed in Fig. 3.4. The implementation is iterating batch-wise over each data point in both datasets. Since the output for every data point can potentially be large, the attack features for a memory-fitting batch is written to an **HDF5** file on disk before calculating the attack features of the next points. This way, the memory consumption during the creation of attack data can be held to a minimum. Additionally, saving the attack features to a file on disk allows for easy reproducibility of experiments, without expensive gradient calculations for repeated runs. D^{train}_{attack} and D^{test}_{attack} is obtained by combining D^{in}_{target} and D^{in}_{target} by the same amount and then split to a test and train set for the attack model.
The extraction of gradients of all hidden layers quickly becomes very intensive

w.r.t. the required disc space. That is due to the attack feature extraction generating a multiplicity of values. Imagine a target model DNN architecture of $[600, 1024, 512, 256, 128, 100]$, that yields an amount of 2620 neurons and more than 1.3 million weights. Furthermore assuming Float64 datatype that would result into a disk space requirement with at least 10 MB for a single data point. As an example: assuming 70000 data points in our dataset and $T = 3$ captured target models the disk requirement would be 2100000 MB or 2.1 TB.
Experiment have shown, that it is sufficient if only the last layer is used to generate attack features. Making use of previous layers, as well, only marginally ($< 1\%$) improved the attack accuracy. Therefore in this thesis, only the last two layers of T target models are used to train the attack model.

Attribute Inference

An attribute inference attack tries to infer the possible value of a unknown sensitive attribute. In this thesis, the membership inference approach from Nasr et al. [9] is extended to allow attribute inference attacks. Generally, membership inference is a generalization that can be used to infer whether an individual x with different attributes $\pi_i(x)$ where i is the sensitive attribute index and $\pi_i(x) \in 0, 1$, were member of the training set or not.
In this work, the membership inference attack model is trained on binary valued datasets. The dataset $\mathcal{D}_{train}^{target}$ is mirrored and the attribute that is being inferred at index i is flipped for every individual $x \in \mathcal{D}_{train}^{target}$. This results in $\mathcal{D}_{target}^{in}$ for the data with the correct attribute and $\mathcal{D}_{target}^{out}$ containing the data with the flipped a.k.a. wrong attributes. Thus, for every attacked attribute at index i there needs to be a MIA model and the corresponding attack features to distinguish between whether $x \in \mathcal{D}_{target}^{in}$ or $\hat{x} \in \mathcal{D}_{target}^{out}$ is more likely.
For every dataset at least 10% of the indices are attacked to get an fair estimation of the true attacker. Moreover, the same target models are used as in the membership inference experiments. Attribute inference constitutes a more realistic and problematic attack technique, because the adversarial knowledge about a sensitive attribute is often worse than the knowledge of sensitive membership. Due to that reason, we consider for an equal MI and AI attack accuracy, the AI attack to be much worse.

3.4 Learning under Noise

In the next sections the actual realization of DP for the following experiments is explained.

LDP

In this work, Local Differential Privacy is used to perturb the sensitive training data for the target model. The entire dataset is perturbed using either a DP Randomized Response mechanism, to flip Boolean values with a certain probability or by applying noise from a Laplacian distribution yielding a Laplace Mechanism. The latter is used for perturbing scalar values, like pixels. That way, the entire dataset exists in a perturbed and an original variant, so that each data owner can train on their perturbed data while the attacker uses the original data. Intuitively, this is because we want the attacker to learn to infer data from the original training dataset rather than the noisy one. This implies, that the attacker has a certain amount of the original unperturbed data and a target model that was trained on noisy data.

CDP

Central Differential Privacy is consumed from the Tensorflow Privacy library. This library provides noisy optimizers which alter traditional optimizers by introducing noise sampled from a Gaussian distribution on to the gradients during training. This effectively implements a Gaussian DP-Mechanism. The training occurs on the original data, but due to the noisy gradient updates, the training of the target model usually takes longer.
When using CDP, the effective privacy loss is measured using the Renyi-differential privacy moments accountant as described in the theoretical foundations. In practice, the RDP is calculated w.r.t. to a fixed set of orders α for a Gaussian mechanism by taking noise parameter σ and the sample-ratio, i.e. the amount of times a certain individual from a dataset has been observed, into account. ϵ can be calculated for a given target δ yielding (ϵ, δ)-DP. To be able to conveniently plot the results of the experiments, in this thesis, the calculated ϵ values are averaged over all identical experiments. Another way of aggregating the ϵ values under composition and for multiple identical experiments is taking the maximum value $max(\vec{\epsilon})$, however the difference between those two methods is diminishing.

Hyperparameter Search

Hyperparameter search is the process of finding the best parameter combination in a potentially large parameter search space. The goal is to minimize an objective function f for a certain set of parameters p:

$$\arg\min_{\vec{p}} f(\vec{p}) \tag{3.2}$$

In this work, contrary to Nasr et al., hyperparameter search, particularly *Gaussian Process* (GP) optimization, is used to find optimal hyperparameters to train the attack model. The attack models of the corresponding target model that are not using DP at all, and those of the target model that use CDP with different noise levels are optimized with a search space containing a continuous learning rate range of $[0, 00005, 0, 003]$, different batch sizes $\{64, 128, 256\}$ and a binary value telling whether *Batch Normalization* is applied or not. Compared to a classical grid search, the Gaussian Process' runtime is restricted by the amount of **calls**, which the process is allowed to make in order to test certain parameter combinations. This particularly makes sense, if the search space is continuous or very large. Here, the GP optimizer, tries to minimize the negative membership advantage of the currently tested attack model. Using an exploration-vs-exploitation trade-off and Bayesian inference, it finds the next potentially suiting parameter candidate [34].

3.5 Approach

The evaluation of the trade-off between utility and privacy w.r.t. different parameter ϵ is done by attacking several ANN classifiers for different datasets. The target model as well as the classification tasks differ in output dimension, training data distribution and ANN complexity. The target models are trained using FL without DP, with LDP and CDP and different noise levels respectively. To measure the privacy, the models are being attacked using the global and local attack scenario. Every experiment is repeated three to five times. Then the results are measured to obtain possible insights.
Wu et al. [30] state, that the convexity of a loss function for a ML model under FL differs when applying ϵ-CDP. They even suggest a particular learning rate and iteration parameter, which will reduce the performance difference between a model trained using DP and the same model in absence of DP to a minimum. This, however, is only suitable for a cost function which *meets the assumptions of smoothness, strong convexity, and Lipschitz-continuity of the gradient* [30]. However, cost functions with such best-case assumptions

are relatively rare in practice. Nasr et al. [9], in contrast, train for a certain amount of epochs and take the best performing model for both target and attacker. In this thesis *early stopping* is used extensively. This gives a much more realistic scenario. Though this approach leads to several consequences in the experimental comparability of the results from Nasr et al. and those from Bernau et al. [5].
For fixing parameter T, Nasr et al. state, that they found out, that stacking attack features from $T = 5$ models from non-consecutive epochs, yields the best result. Furthermore they point out, that using latter epochs yields even better results, because the earlier epochs only learn general information without much membership information. In this work the attack model performance could vary depending on how long the target model is trained. Under certain circumstances some experiment repetitions could lead to more or less observed communication rounds. Therefore, the attacker can use more or less observed target models from different epochs which could lead to a worse or better attack accuracy. Moreover using RDP to calculate ϵ under composition could vary as well, since RDP highly depends on how often the target model observes a certain perturbed individual from a dataset during training. This effectively leads to different ϵ values for some repeated experiments with early stopping. Each experiment is repeated five times with the same overall configuration but with different random seeds. All experimental results are averaged and presented with standard deviation. The precision-recall curves take the thresholds of every identical experiment into account and the ROC curves are interpolated among all identical experiments. For every dataset, there is an experiment for a global and local attacker without DP and with CDP and LDP. CDP and LDP are tested using five different ϵ values.

3.6 Attack Evaluation Metrics

In this work, the attack model is evaluated not only by using the accuracy but a few different metrics. The attack evaluation methodology is largely based on the methodology used by Bernau et al. [5]. Since the adversary constitutes a binary classifier, the overall performance for a certain test run, can be measured using the amount of correctly identified members called True Positives (TP), and the True Negatives (TN), i.e. the correctly identified non-members. Consequently, False Positives (FP) are non-members who falsely got classified as members, also called Type-I-Error, and False Negatives (FN), who are misidentified non-members, also called Type-II-Error.

Test Accuracy

Accuracy is defined as:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Since the attack model makes binary decisions, random guessing would achieve a 0.5 accuracy. Therefore, it is fair to assume a 0.5 worst-case or baseline accuracy. A perfect performance would reach an accuracy of 1.0, i.e. identifying every member and non-member correctly. The Accuracy is also used as measurement for the utility of the target models.

Attack Precision

Precision is defined as :

$$\frac{TP}{TP + FP}$$

The Precision expresses what proportion of the positives are TP. Precision is arguably an important performance metric for the attack model, since it expresses how reliable member identification is. An attacker tries to focus on identifying TP and can mostly ignore TN. Just like the Accuracy, the baseline is 0.5.

Attack Recall

The recall metric represents the proportion of actual member who have been classified correctly:

$$\frac{TP}{TP + FN}$$

Ideally, an attack model should score high in both precision and recall, to accomplish a successful inference attack. The baseline for Recall is 0.

False Positive Rate

The FPR is the opposite of the Recall and is used in the ROC curve, it yields the ratio between falsely identified individuals over all individuals.

$$\frac{FP}{FP + TN}$$

Membership Advantage

Yeom et al. [16] introduce a metric called *Membership Advantage*. This metric measures how well an adversary can distinguish between $z \sim D'$ and $z \sim D$ after being given the model. It can be calculated by:

$$Recall - FPR$$

If the attacker randomly guesses, this metric yields 0. In this thesis, Membership Advantage is used to evaluate the attack model during the hyperparameter search. Yeom et al. proved, that the membership advantage is bound by $e^{\epsilon} - 1$.

Since the attack model's last layer has a sigmoid activation function, instead of a label corresponding to the input, we receive a probability. Whether a member belongs to a certain class depends on a threshold for the classification probability. If the threshold for being classified as "in" is set high, we receive a high precision, because we are certain that the input data belongs to a class, consequently we receive a small recall, because we classify more data as False Negatives. Vice versa, if we set the threshold low, we receive a low precision, since we are labeling more data as being "in", but a high recall, because we have a lot less False Negatives. Therefore, it is important to set the threshold according to the needs and moreover to balance recall and precision, so that both are high. For that particular reason, it is crucial to evaluate the binary classifier for a multitude of different thresholds [35]. Moreover, it is not suitable to only use a single-threshold measurement like Membership Advantage or accuracy to grasp the attackers full capabilities. In this thesis and for above reasons, the *Receiver Operating Characteristics* (ROC)- and *Precision-Recall*-Curves [36] are proposed to better evaluate an attacker.

Receiver Operating Characteristic Curve

The ROC-Curve plots the Recall on the y-axis and the FPR on the x-axis for different classification thresholds. A straight line from $[0, 0]$ to $[1, 1]$ identifies the baseline. The ROC curve provides an *Area under Curve* (AUC) value, which is a scalar metric indicating the overall models performance for all thresholds.

Precision-Recall Curve

The Precision-Recall (PR) Curve plots the Precision on the y-axis and the Recall on the x-axis, for different classification thresholds. The baseline is the same as for the ROC curve: a straight line from $[0, 0]$ to $[1, 1]$. Compared to the ROC-Curve, the PR-Curve has no meaningful AUC value due to the non-monoticity and thus the average weighted precision is used as an alternative measurement.

The last metric, taken from Bernau et al. [5], measures the ratio between target model utility and privacy by taking the target models accuracy and the attack models behavior into account.

Privacy Gain - Performance Loss Ratio

Assuming AUC_{orig} to be the AUC for the attacker when attacking a target model without DP, AUC_ϵ be the AUC for attacking a target model when DP (with a certain ϵ) is applied and AUC_{base} being the baseline of 0.5. And furthermore assuming ACC_{orig} to be the target models accuracy without DP being applied, ACC_ϵ being the target models accuracy when DP (with a certain ϵ) being applied and lastly ACC_{base} being the target models baseline accuracy $1/C$ with C to be the amount of classification classes.
φ measures the Privacy Gain - Performance Loss Ratio:

$$\varphi = \max\left(2, \frac{\max(0, (AUC_{orig} - AUC_\epsilon) \cdot (ACC_{orig} - ACC_{base})}{\max(0, (ACC_{orig} - ACC_\epsilon) \cdot (AUC_{orig} - AUC_{base})}\right)$$

This metric yields two possible outcomes: $0 < \varphi < 1$, when the loss in test accuracy exceeds the gain in privacy and $1 < \varphi < 2$ when the relative gain in privacy exceeds the relative loss in accuracy. Therefore, an optimal trade-off outcome would be a value close to 2 [5].

Experiments

This chapter presents and analyzes the experiments which were performed throughout this thesis. First, the experimental environment and setup is laid out and afterwards, the datasets and corresponding target models are introduced. Finally, the experimental results for the target model under noise application and the results for the inference attacks using Membership and Attribute Inference, are shown.

4.1 Experimental Setup

This section describes the general experimental setup, which is used to perform the experiments. The hardware and the software setups are briefly described.

Hardware

For the experiments, AWS EC2 instances of type "c5d.9xlarge" with 36 vCPUs and 72GB RAM are used for datasets like *Texas Hospital Stay* or *Purchases Shopping Carts*. The larger EC2 instances "c5d.24xlarge" with 96 vCPUs and 192GB RAM are used for the *Labeled Faces in the Wild* (LFW) dataset. Both instance types are compute optimized. The GPU acceleration brought little to no benefit, which is due to the federated protocol, the network I/O and the reading of vast amount of data from the disk into the memory, causing the actual bottlenecks.

Software

For the experiments in this thesis Tensorflow (version 1.15.2) is used through the high-level Keras API (version 2.2.4). To communicate over network

within the federated framework introduced in the last chapter, Socketio is used. The DP-optimizer is taken from the publicly available TensorFlow-Privacy package [37]. Nasr et al. [9] state that they have used PyTorch for the implementation of their MI attack. However, since there is already a pre-existing MI framework developed within SAP SE, which makes use of TensorFlow and Keras, PyTorch is not further considered. At the time of writing, Nasr et al. have not released their code and there is no publicly available implementation of their attack. This is why the implementation for performing experiments is done independently and solely based on the description given in the research paper of Nasr et al. to maximize reproducibility of the code, the MI attack is implemented as a Keras custom layer and will be provided on GitHub. This allows building an attack model in a descriptive manner using a sequential Keras model.

4.2 Datasets

This section introduces the datasets which are used in the experiments. General characteristics of the datasets are described, e.g., size, dimensionality, and domain. Moreover, the target model architecture differs depending on the dataset. Therefore, the corresponding model architectures are also introduced. Finally, the choice for hyperparameters for target model training without DP (NoDP), with LDP and with CDP are laid out.

4.2.1 Purchases Shopping Carts

The dataset "Purchases Shopping Carts" [38], conveniently referred to as "Purchases" dataset, is provided by Shokri et al. [2] and is based on the Kaggle dataset "Aquire valued shoppers". The dataset contains the shopping histories of individuals. Each entry consists of 600 binary values indicating whether a particular item was bought by an individual or not. The dataset has more than 200000 individuals. The individuals were separated using a k-means clustering with $k = [10, 20, 50, 100]$. In Histogram 4.1 the class frequencies of the Purchases dataset is shown separated in 100 k-means cluster. As seen the dataset is quite unbalanced and some classes are underrepresented compared to others.

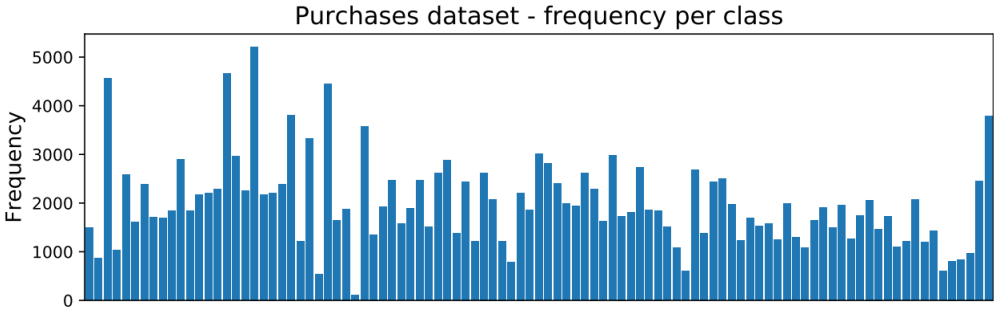


Figure 4.1: Class frequencies in the "Purchases100" dataset.

Target model architecture For the experiments on the Purchases dataset, the target model constitutes a Fully Connected Network with six layers containing

$$600, 1024, 512, 256, 128, 100$$

neurons. With the first layer being the input layer and the last layer denoting the 100 output neurons. Beware, that the last layer dimension can vary, depending on the class configuration, for instance $10, 20, 50$ is used in some of the experiments as well. *Tanh* is used as activation function in the hidden layers and *Softmax* for the output layer. *Categorical cross-entropy* is used as a loss function. The target models objective is, for any individual from the dataset, to associate a label that corresponds to his k-means cluster. The federated averaging hyperparameters are fixed accordingly for all datasets:

$$E = 1, K = 4, C = 1$$

These parameters are supposedly the same Nasr et al. used for their experiments. E denotes the amount of local epochs per data owner, K the amount of overall clients and finally C describes how many of the K clients train in parallel (1 means all).

No-DP training The models are trained on a training set of size 20000 and test set of size 50000, this corresponds to the setting of Nasr et al. [9]. The batch-size is set to 100 and the model is trained using the *Adam* optimizer with a learning-rate set to 0.001.

LDP training The training hyperparameters are identical to the No-DP parameters. The only difference is, that the model is trained on a noisy dataset. The binary values of each data point are retained with a

probability of $\frac{e^{\epsilon}-1}{e^{\epsilon}+1}$. For the experiments the following privacy parameters are used:

$$\epsilon_i \in \{3, 2, 1, 0.5, 0.1\}$$

with retention probabilities of:

$$0.9\%, 0.76\%, 0.46\%, 0.24\%, 0.12\%, 0.05\%$$

The LDP mechanism used for the Purchases dataset uses a form of randomized response. Noise is applied only to $\mathcal{D}^{train}_{target_k}$, whereas $\mathcal{D}^{test}_{target_k}$ remains untouched.

CDP training For CDP training, the *DPAdam* optimizer is used. The hyperparameters are identical to the LDP and No-DP ones. The noise clipping $\mathcal{C}$ impacts the performance of the trained model, as the magnitude of $\mathcal{C}$ influences the amount of noise added to the gradients during training. A clipping of $\mathcal{C} = 2$ produces the best result and is hence used for the experiments with noise multiplier $z \in \{0.5, 2, 4, 6, 16\}$. Because DPAdam uses a Gaussian DP mechanism, parameter δ needs to be fixed beforehand. In practice it is common to set $\delta = \frac{1}{|\mathcal{D}^{train}_{target}|}$ yielding $\delta = 5 \cdot 10^{-5}$. The resulting ϵ values using RDP composition are:

$$\epsilon \in \{75.84, 5.26, 2.35, 1.52, 0.54\}$$

4.2.2 Texas Hospital Stays

The "Texas Hospital Stays" dataset is taken from Shokri et al. [2] and is based upon "Hospital Discharge Data" [39] from the Texas Department of State Health Services. In this thesis, the dataset is referred to as "Texas" dataset for convenience reasons. The dataset has 67333 entities consisting of a binary feature vector of length 6170 and is divided into $100, 150, 200, 300$ classes with the same technique as the Purchases dataset. Each binary value represents an injury, a medical procedure or binary encoded information about the individual, such as gender, age or region.

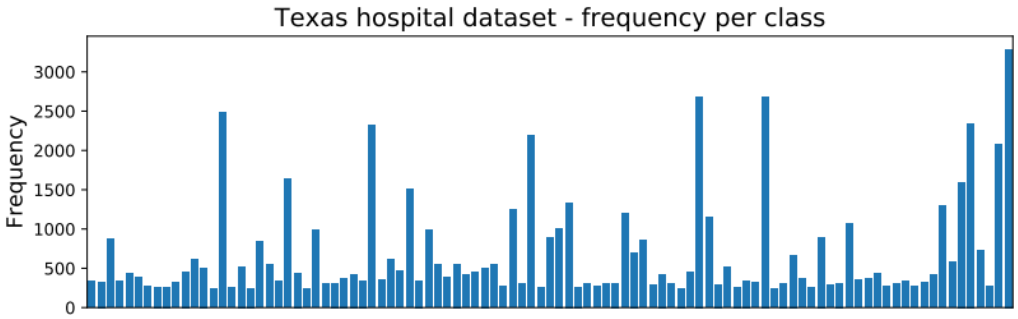


Figure 4.2: Class frequencies in the "Texas100" dataset.

Histogram 4.2 shows the class frequencies for Texas100. Compared to the Purchases dataset, the Texas dataset is even more unbalanced.

Target model architecture The target model for the Texas100, Texas150, Texas200 and Texas300 datasets classifying $100, 150, 200, 300$ labels consists of a fully connected network similar to the Purchases target models. The number of Neurons for the exemplary case of Texas100 are:

$$6169, 1024, 512, 256, 128, 100$$

Tanh activation function in the hidden layers and *Softmax* for the output layer is used. *Categorical cross-entropy* is used as loss function. The target model architecture and configuration are chosen in accordance with the experiments of Nasr et al. [9] to ensure comparability of the results.

No-DP training The models are trained on a training set of size 10000 and a test set of size 57300. Batch-size is set to 100 and the model is trained using the *Adam* optimizer. The learning rate is set to 0.001.

LDP training All training hyperparameters from the No-DP training remain unchanged in the LDP experiments. Following privacy parameters are assessed in the LDP case:

$$\epsilon_i \in \{3, 2, 1, 0.5, 0.1\}$$

The LDP mechanism is equivalent to the one from the Purchases target model. Randomized Response is applied to $D^{train}_{target_k}$, whereas $D^{test}_{target_k}$ has the original values.

CDP training Using CDP, the learning rate is set to 0.0001, as it converged faster in our experiments. The batch-size is kept identical at 100. The clipping norm was found using hyperparameter search and is set to $\mathcal{C} = 2$. The noise multipliers used are: $z \in \{0.5, 2, 4, 6, 16\}$. ϵ parameter is evaluated using $\delta = 1 \cdot 10^{-4}$. The resulting ϵ values are:

$$\epsilon \in \{74.15, 3.8, 1.72, 1.19, 0.44\}$$

4.2.3 Labeled Faces in the Wild

The dataset "Labeled Faces in the Wild" (LFW) is a dataset often used as benchmark dataset for facial recognition and verification. It contains 2773 images of faces of either 20 or 50 individuals with a resolution of 255×255 pixel per image. Some samples taken from the dataset can be seen in Figure 4.3.

Figure 4.3: Some samples taken from the "LFW" dataset.

As seen in Histogram 4.4, the dataset is unbalanced, just like the other datasets, but in contrast, one single class is strongly overrepresented. The motivation for using a dataset containing faces of real people is due to the increasing use of facial recognition algorithms. Inference attacks on this dataset constitute realistic scenarios.

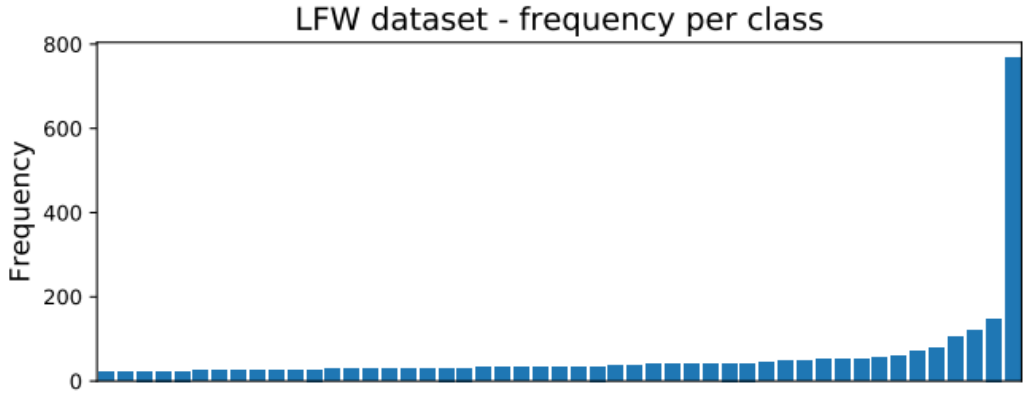


Figure 4.4: Class frequencies in the "LFW50" dataset.

Target model architecture For the LFW target model, a pre-trained "VGG-Very-Deep-16" CNN [40] is used. The target model should label each image to its corresponding person. The overall architecture of the VGG-Very-Deep-16 CNN is shown in Figure 4.5.

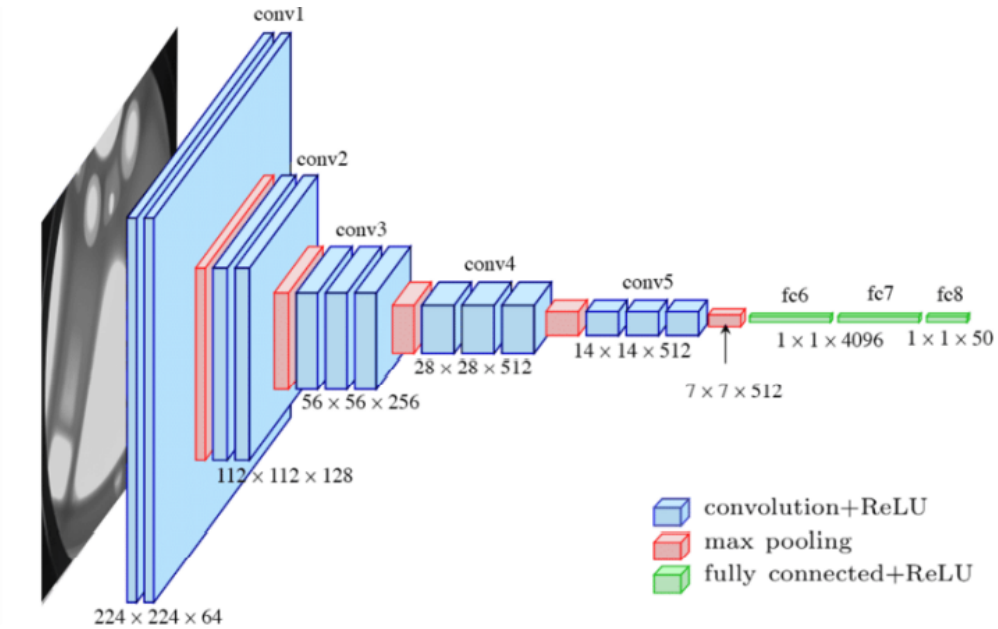


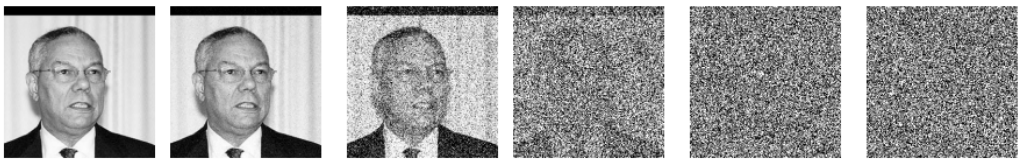
Figure 4.5: VGG architecture for LFW50 target model[41].

The weights of the pre-trained VGG CNN (blue layers in Figure 4.5) are fixed, such that only the last FCN layers' weights (green layers in Figure 4.5) are updated during training. This, additionally, is a huge performance boost in the Federated Learning case, because a much smaller weight matrix has to be propagated through the network. The fully connected network uses *ReLu* activation functions, except for the last one, which uses a *Softmax* activation function. *Categorical cross-entropy* is used as loss function. The images from the LFW dataset are pre-processed before they are passed to the target model. Each image is first converted from an *RGB* to a gray-scale image. Finally, the values which range from 0 to 255 are normalized to an interval between 0 and 1.

No-DP training The models are trained on a training set of size 1000 and a test set of size 1700. This train-test split is also used in LDP and CDP training. The batch-size is set to 64. To train the target model, *Adam* optimizer is used. Finally, the learning-rate is set to 0.001 for the experiments.

LDP training LDP is applied to the images using a mechanism that is also used by Bernau et al. [5] and is based on a mechanism proposed by Fan [42]. The technique is similar to a ϵ-DP Laplace Mech-

anism as introduced in the theoretical foundations Chapter 2. A randomizer applies noise to each pixel of the image following a Laplace distribution of scale $\lambda = \frac{255 \cdot m}{b^2 \cdot \epsilon}$. The parameter m represents the neighborhood, as global connectivity analogy, of a pixel and is set to $m = \sqrt{250 \cdot 250} = 250$ for all LDP experiments.
The neighborhood parameter brings its own trade-off, taken from Bernau et al.: "Full neighborhood for an image dataset would require that any picture can become any other picture. In general, providing DP within a large neighborhood will require high ϵ values to retain meaningful image structure. High privacy will result in random black and white images.[5]"
In other words, a larger neighborhood will perturb the images more globally and result in either worse image quality or lower privacy guaranties.



(a) original (b) ϵ_i=10 000 (c) ϵ_i=1000 (d) ϵ_i=100 (e) ϵ_i=10 (f) ϵ_i=1

Figure 4.6: Example images from LFW with different levels of noise applied using LDP mechanism.

Images with different values for ϵ_i are shown in Figure 4.6. To the human eye, a noise level that corresponds to $\epsilon_i = 1000$ still allows to recognize and even identify a person. On $\epsilon_i = 100$, a silhouette can still be seen but makes identification by human hardly possible.

CDP training Using CDP, the target model is trained with a lower learning-rate of 0.0001 and smaller batch-size of 32. The clipping norm is fixed to $\mathcal{C} = 2$, the noise multipliers used are $z \in \{0.5, 2, 4, 6, 16\}$ and $\delta = 1 \cdot 10^{-3}$ using the formula introduced earlier. The resulting ϵ values are:

$$\epsilon \in \{102.52, 5.27, 2.33, 1.45, 0.54\}$$

4.3 Target Model Results

In the following section, the experimental results of the target model classifiers for the introduced datasets are shown. The experiments show

the accuracy of each classifier and the same target models after applying LDP and CDP with different noise levels as explained in the last section "Datasets" 4.2. For every dataset the plots show the validation and training accuracy for the classifier w.r.t. different noise levels. These models are then used in the next section for the inference attack experiments.
The target models are trained using the Federated Learning framework outlined in Section 3.3. As already discussed the Federated Learning parameters are set to:

$$E = 1, K = 4, C = 1$$

4.3.1 Purchases Shopping Carts

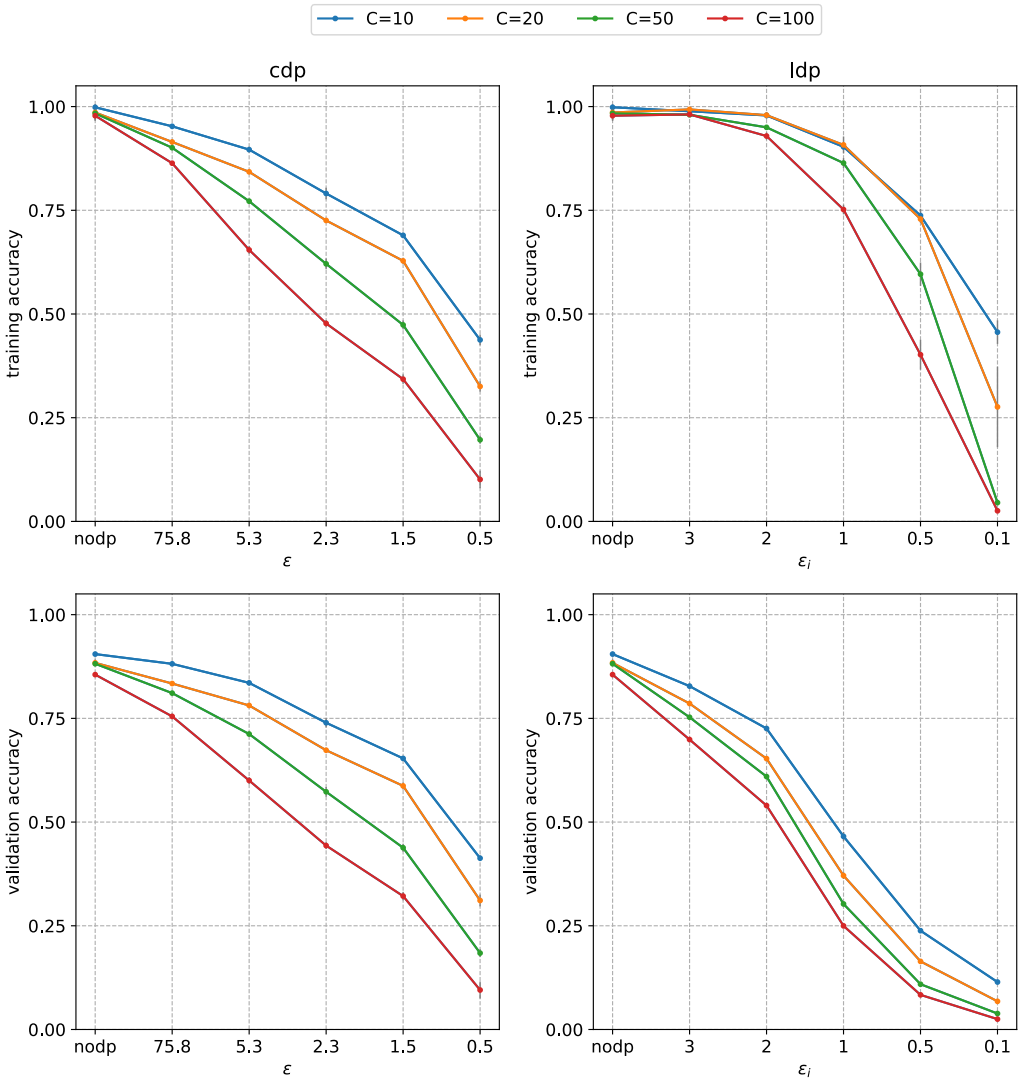


Figure 4.7: Target model performance on Purchases dataset.

The Purchases classifiers in Figure 4.7 show a clear trend in decreasing performance with increasing noise. We can see the four different class configurations with $10, 20, 50$ and 100 labels. The less complex the learning task, the less susceptible it is for increasing noise. The baseline, i.e. the accuracy of a random guess, for every experiment is $\frac{1}{c}$ with c being the amount of output labels. With the CDP-mechanism $\epsilon = 0.54$ and LDP local $\epsilon_i = 0.1$ the models get closest to baseline. The

target model for LDP stays constant until $\epsilon_i = 2$ while using CDP the noise is affecting the target model already with $\epsilon = 75.84$. This is due to the slight stronger privacy guaranty by a global epsilon.
Furthermore, there is a larger train-test-gap when applying LDP. This suggests a more successful attack later on. The target model NoDP results are equal to Nasr et al. [9], they achieved a train accuracy of 100% and a test accuracy of 77.5% for Purchases100, while in our experiments the validation accuracy is slightly better with 79.5%. Which consequently means, that, in this work, the train-test-gap is slightly smaller and the attack could be performing slightly worse.

4.3.2 Texas Hospital Stays

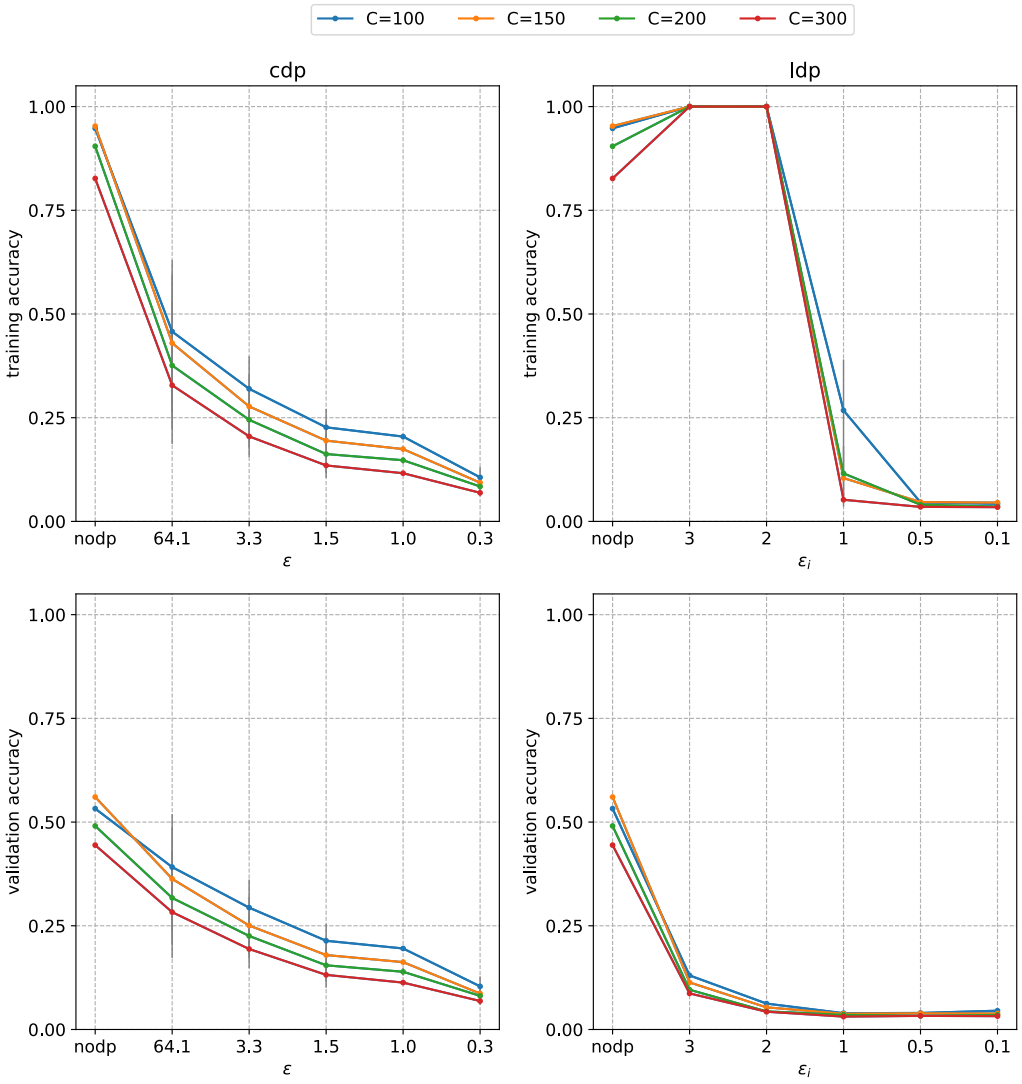


Figure 4.8: Target model performance on Texas dataset.

In Figure 4.8 the target models classifier performance can be seen for $100, 150, 200$ and 300 output dimensions. The LDP mechanism seems to hold the training performance a little longer constant around 1.0 compared to CDP. However the overall validation accuracy's peak is only at 0.5, which is not too bad considering the baseline for Texas300 is $\frac{1}{300}$. The LDP experiments here show a large train-test-gap, compared to the CDP ones, this might suggest a good attacker accuracy.

Interestingly, the train accuracy drops instantly fom 1 to 0 with $\epsilon_i = 2$ to $\epsilon_i = 1$.
Nasr et al. [9] achieved a train accuracy of 95.2% and a test accuracy of 48.5% for Texas100. Our experiments have identical results.

4.3.3 Labeled Faces in the Wild

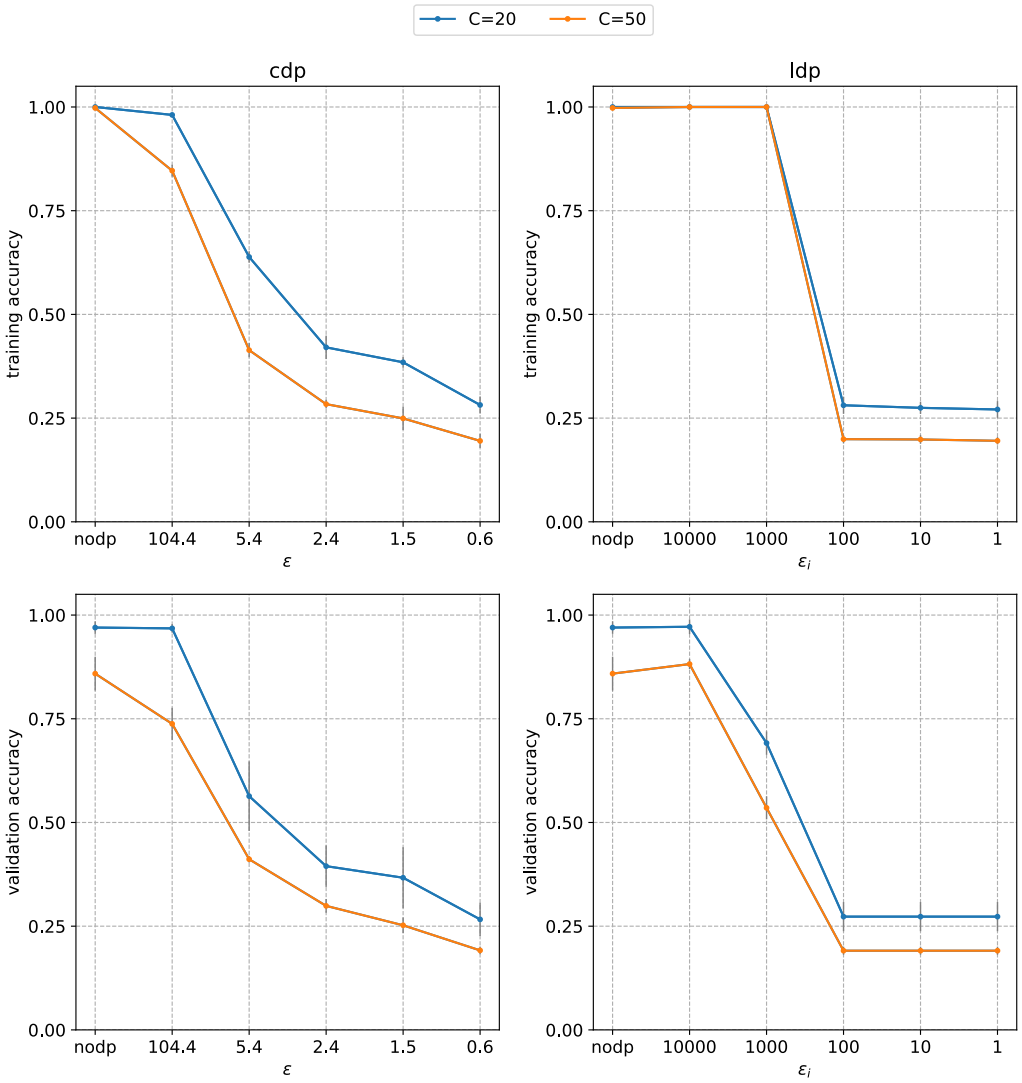


Figure 4.9: Target model performance on LFW dataset.

Lastly, on Figure 4.9, we can see the target model performance for the LFW dataset for 20 and 50 labels respectively. The utility decreases

steeper for the CDP mechanism than for the LDP one. Although, for $e_i = 100$ the loss in utility is about the same as for $e = 2.38$ which shows how hard to compare these two DP applications are. Compared to Bernau et al. [5] our results are identical.

4.4 MI Attack Model Results

In the following subsections the Membership Inference attack results will be presented for each different dataset and with different DP configurations. For each dataset, first, the overall attack performance including accuracy, precision vs recall and privacy-loss-utility trade-off ϕ will be presented for the target models with the different introduced DP mechanisms. Then, exemplary ROC and precision-recall curves will be shown for some particular experiments to demonstrate some threshold depending measurements as described in the Methodology 3. All other experiments can be found in the Appendix. The MI attack model architecture and experimental methodology is explained in Section 3.3.

4.4.1 Purchases Shopping Carts

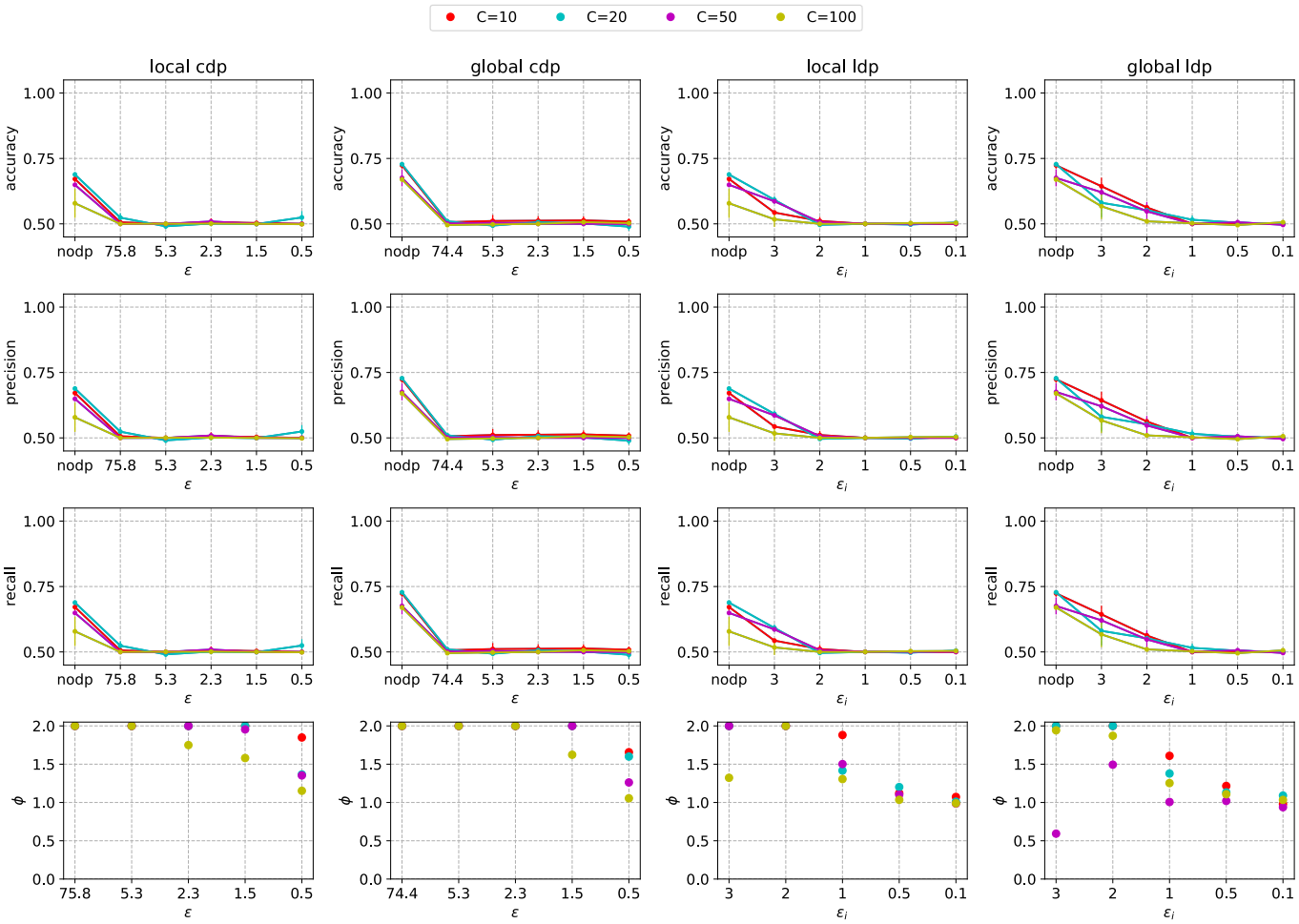


Figure 4.10: MI attack model performance on Purchases dataset.

In Figure 4.10 the attack performance measured in accuracy, precision and recall, and ϕ for the Purchases$[10, 20, 50, 100]$ dataset with different DP configurations can be seen. The figure is separated in four columns, each containing four plots with the results of a global or local attack and either CDP or LDP DP-mechanisms. The last row measures ϕ, i.e. the privacy loss with its relative target model utility. Every drawn point is the mean of multiple experiment repetitions and the gray vertical lines show the standard deviation of the repetitions.
It can be seen, that the attack performance decreases with increasing noise in the target model training. Furthermore, DNN with less complex archi-

tectures, i.e. less output neurons, seem to be more susceptible to the MI attack.

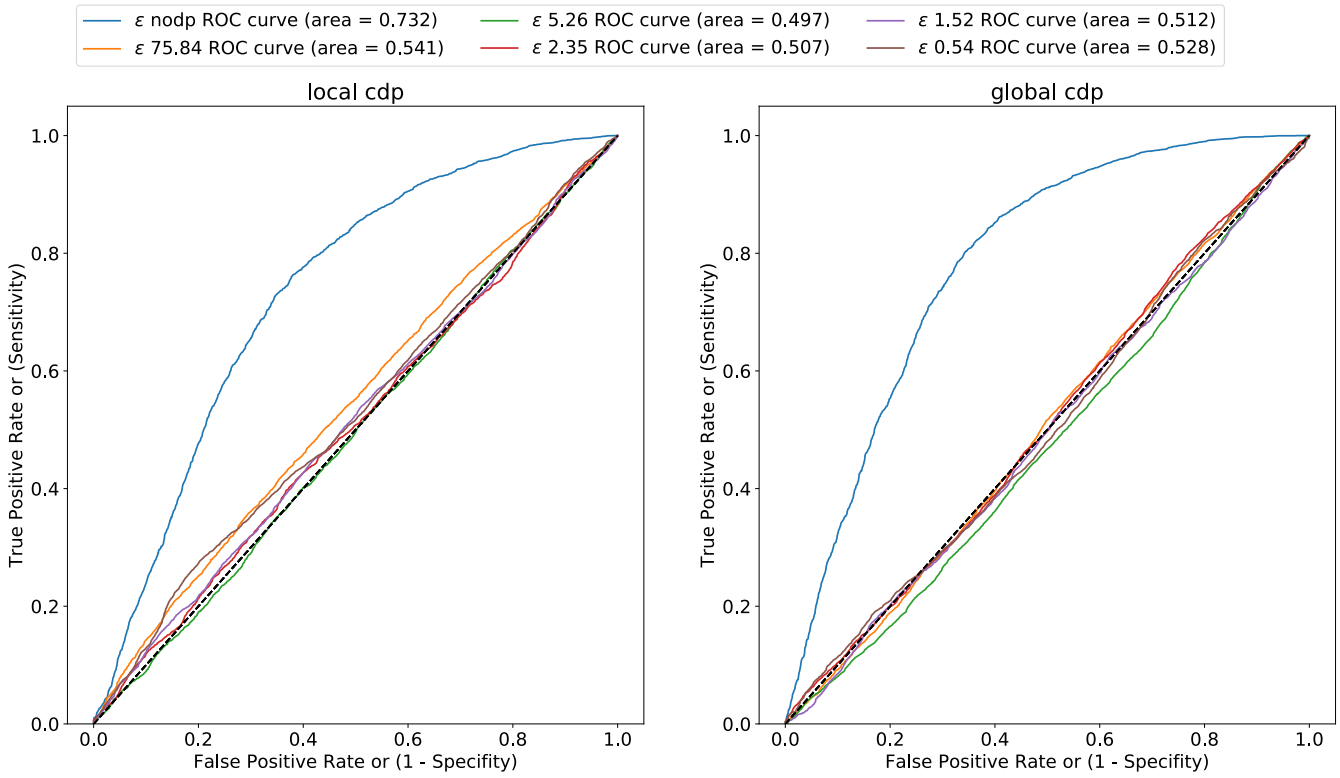
Comparison of global and local attack As expected, it becomes obvious, that the global attacker is much stronger than the local attacker. He achieves an accuracy of nearly 0.75, whereas the local attack only gets close to 0.7. Intuitively, this is because the global attacker accesses the local models of the data owner before they get averaged. Whereas the local attacker has merely access to the aggregated models.

Comparison of DP learning techniques CDP constitutes a strong defense mechanism against MIA for this experiment set. The attack performance drops to baseline even with a relatively high ϵ ($\epsilon = 75.84$). The LDP mechanism shows that for some ϵ_i the attacker is still able to extract membership information, in the local as well as for the global attack. With $\epsilon_i \leq 1$ the attack starts to fail, with this DP parameter, however, the utility of the target model is already at half of the initial utility. For that reason ϕ gets close to 2.0 only for $\epsilon_i > 1$. Although CDP ϵ is a stronger bound than ϵ_i, the target models validation accuracy is better compared to LDP. While the attacker performance on a target model with CDP is instantly at baseline.
Although the CDP ϵ-values are a magnitude of 10 times higher than the LDP ϵ_i values, the overall effect is very similar.

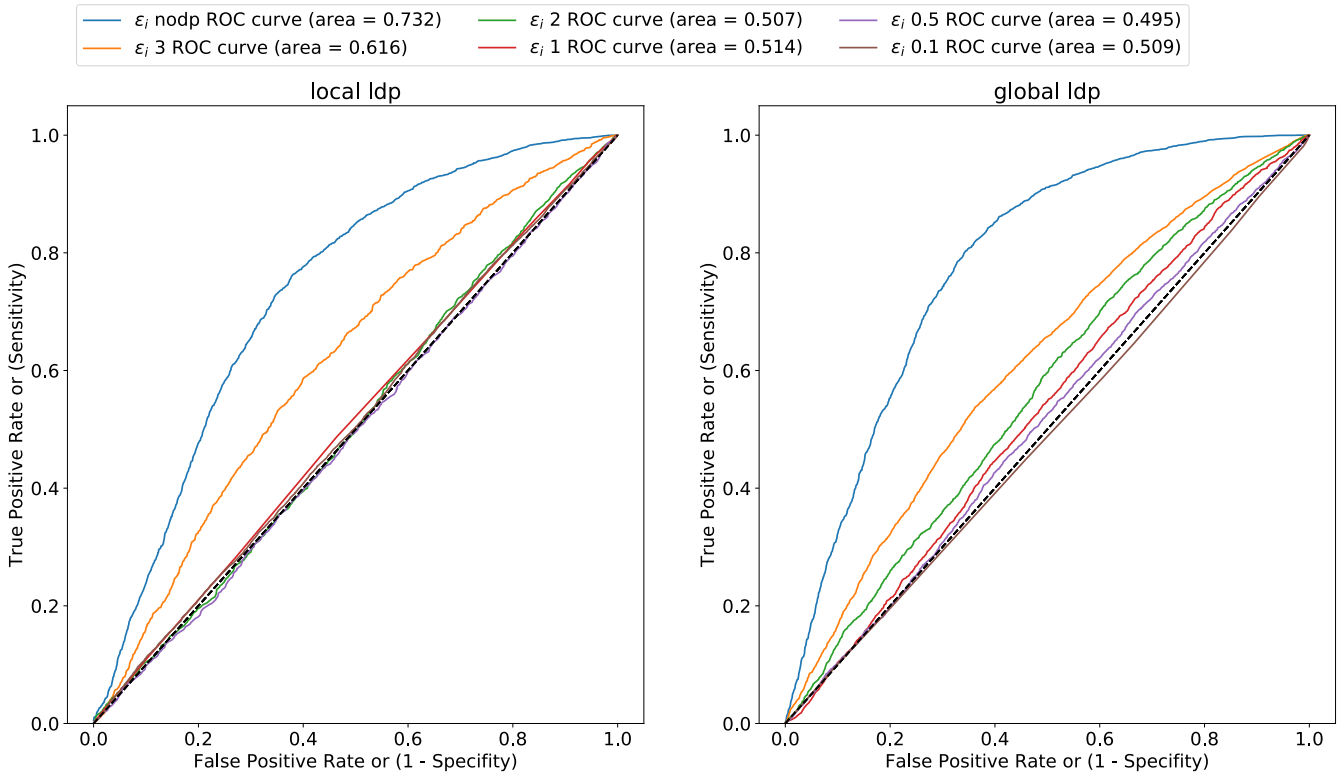
ϕ results With CDP, ϕ gave best results with $\epsilon \in \{75.84, 5.26\}$ for Purchases100, which is due to the attack failing for this class configuration. For Purchases$[10, 20, 50]$ an $\epsilon \in \{2.35, 1.52\}$ seems promising. Using LDP, it demonstrates that $\epsilon_i \in \{3, 2\}$ yields the best outcome against local and global attacks, although it does not result in full defense against MIA. Overall a CDP strategy with $\epsilon \in [75.84, 1.52]$ achieves the best utility-privacy trade-off for both target model configurations and both DP learning mechanisms.

In Figure 4.11 the ROC curves for purchases20 are shown. Purchases20 achieved the highest MI accuracy and is therefore the most interesting configuration in this case. Every colored line depicts an ϵ and every plot a combination of local/global attack and CDP/LDP trained target model.
We can see that for different thresholds, MI on NoDP is always outperforming attacks on DP target models. However, it is much easier now to distinguish, that LDP might not preserve privacy as good as CDP does. The same insights reveals Figure 4.12 using the precision-recall-curve. There is no threshold for the attacker that results in any meaningful success. Furthermore, it is now more apparent, that the global adversary constitutes a

stronger attacker than the local one.
The global attack from Nasr et al. [9] performed better for Purchases100, they demonstrate a MI accuracy of 72.4% while in this work only a MI accuracy of 63% could be achieved on Purchases100 (NoDP). However attacks on less complex Purchases configurations happen to be more successful.

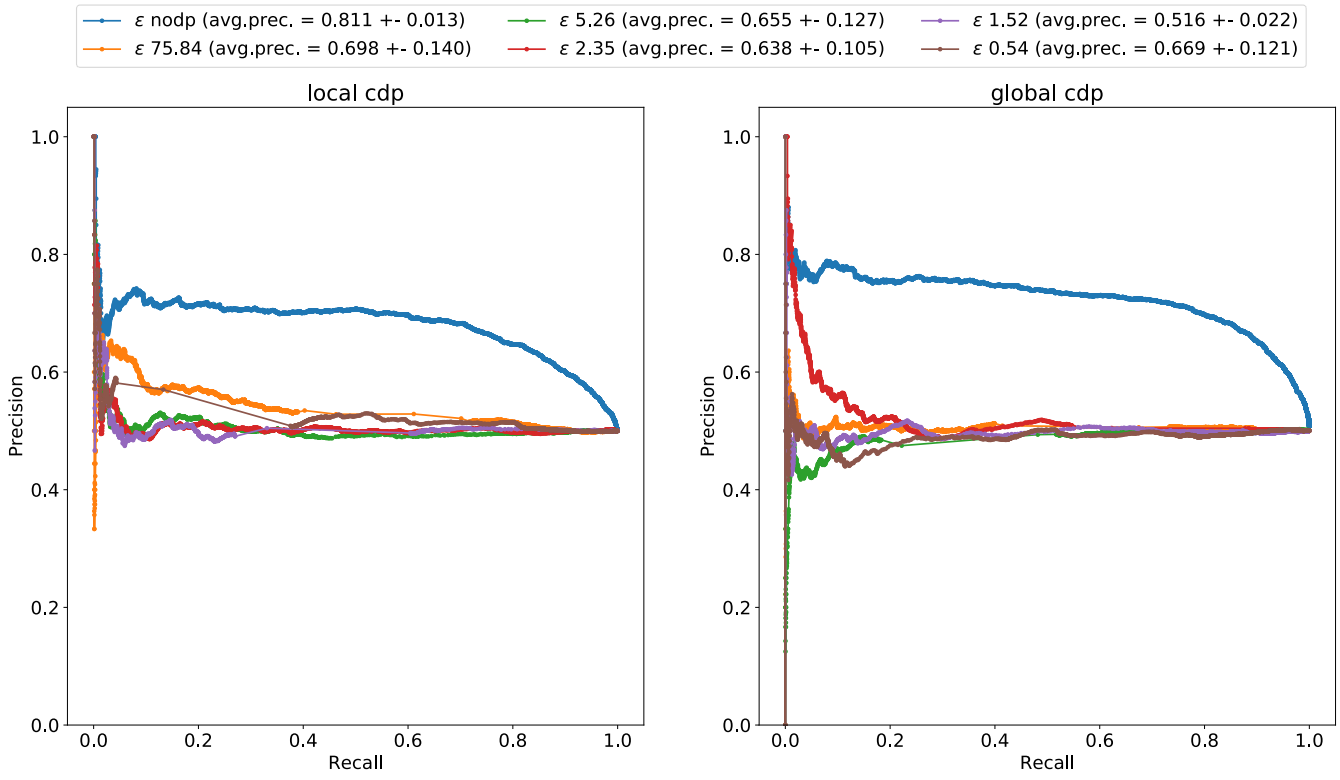


(a)

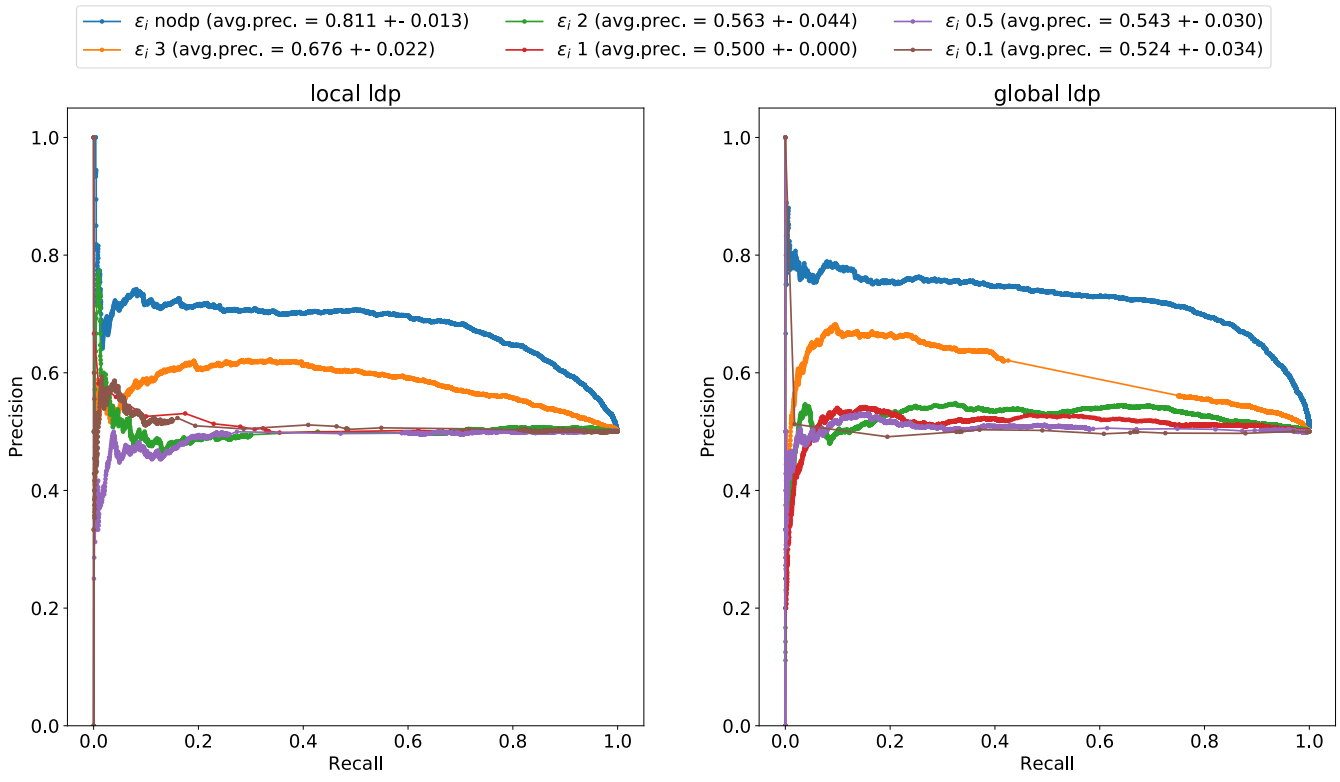


(b)

Figure 4.11: ROC Curve for MIA on Purchases20 dataset



(a)



(b)

Figure 4.12: Precision-Recall Curve for MIA on Purchases20 dataset

4.4.2 Texas Hospital Stays

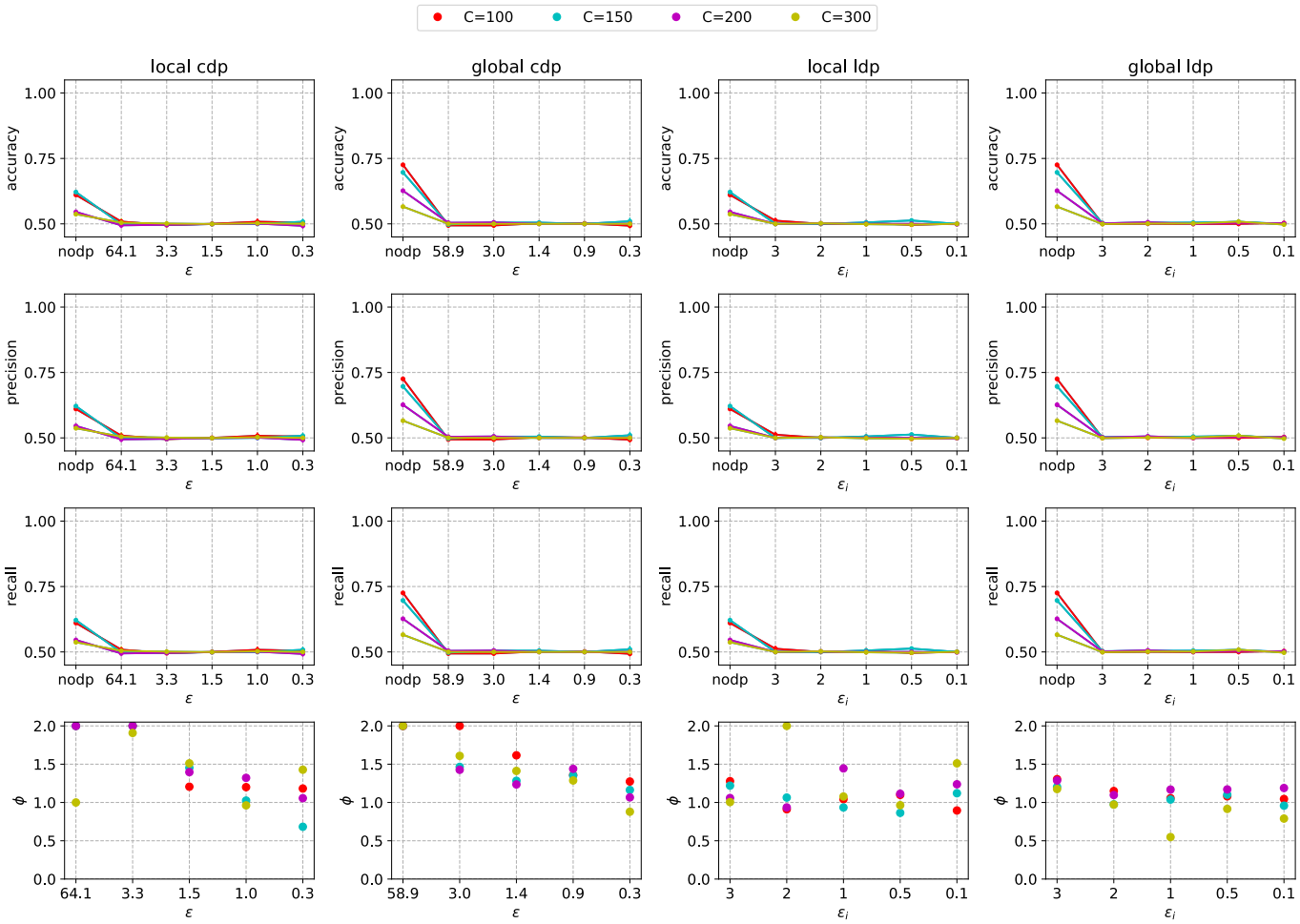


Figure 4.13: MI attack model performance on Texas dataset.

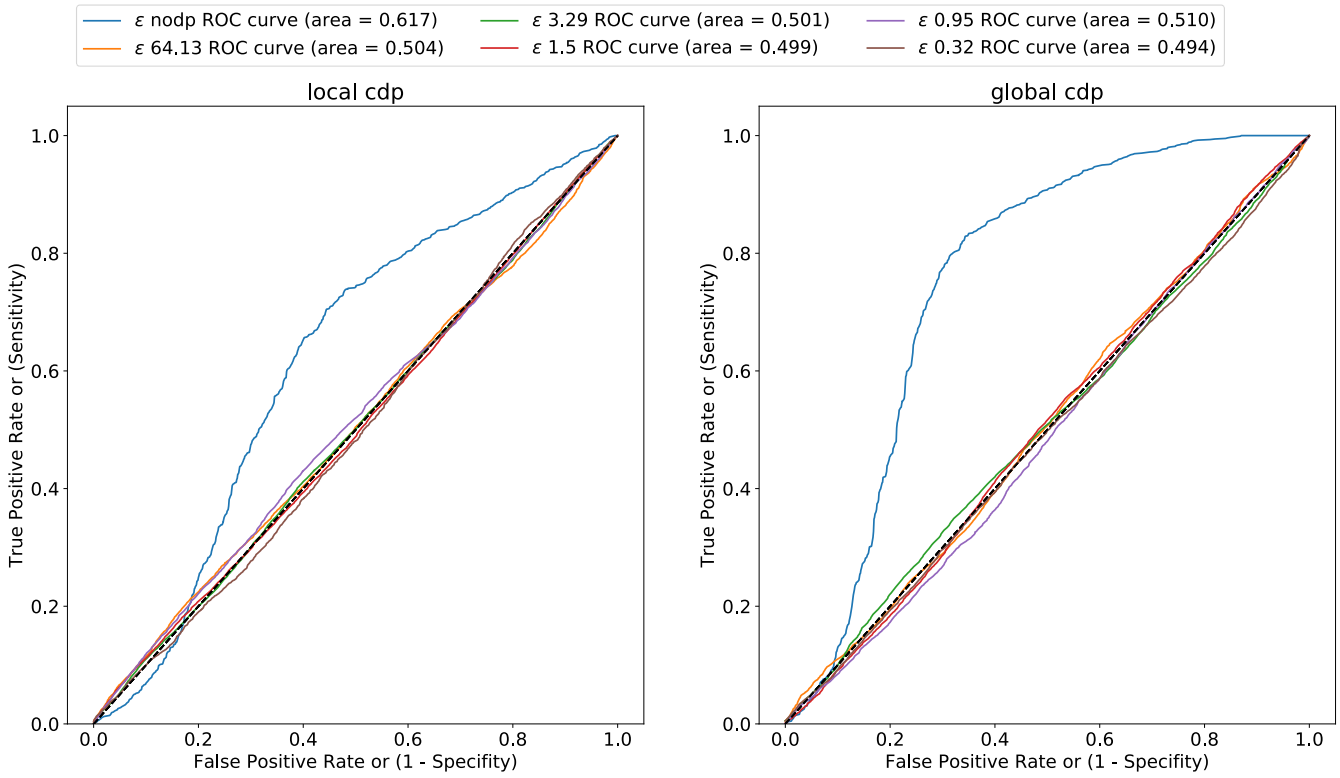
In Figure 4.13, the attack performance on the Texas[100, 150, 200, 300] dataset is evaluated. Just as in the Purchases experiments, accuracy, precision, recall and ϕ can be seen. In this experiment the utility as well as the attack performance already declines with high ϵ both for CDP and LDP and both for the global and local attack. Nasr et al. [9] achieved a global attack accuracy of 66.4% and a local accuracy of 62.4% for Texas100. In this work a global attack accuracy for Texas100 (NodP) of 74% could be achieved while the local attack performance is identical to the one from Nasr et al. Note that the retention probabilities and Federated Learning parameters stay the same as in the Purchases experiments.

Comparison of global and local attack Just like in the Purchases experiments, the global attacker outperforms the local attacker. While the global attack achieves a accuracy of nearly 75% the local attacker is at around 60%.

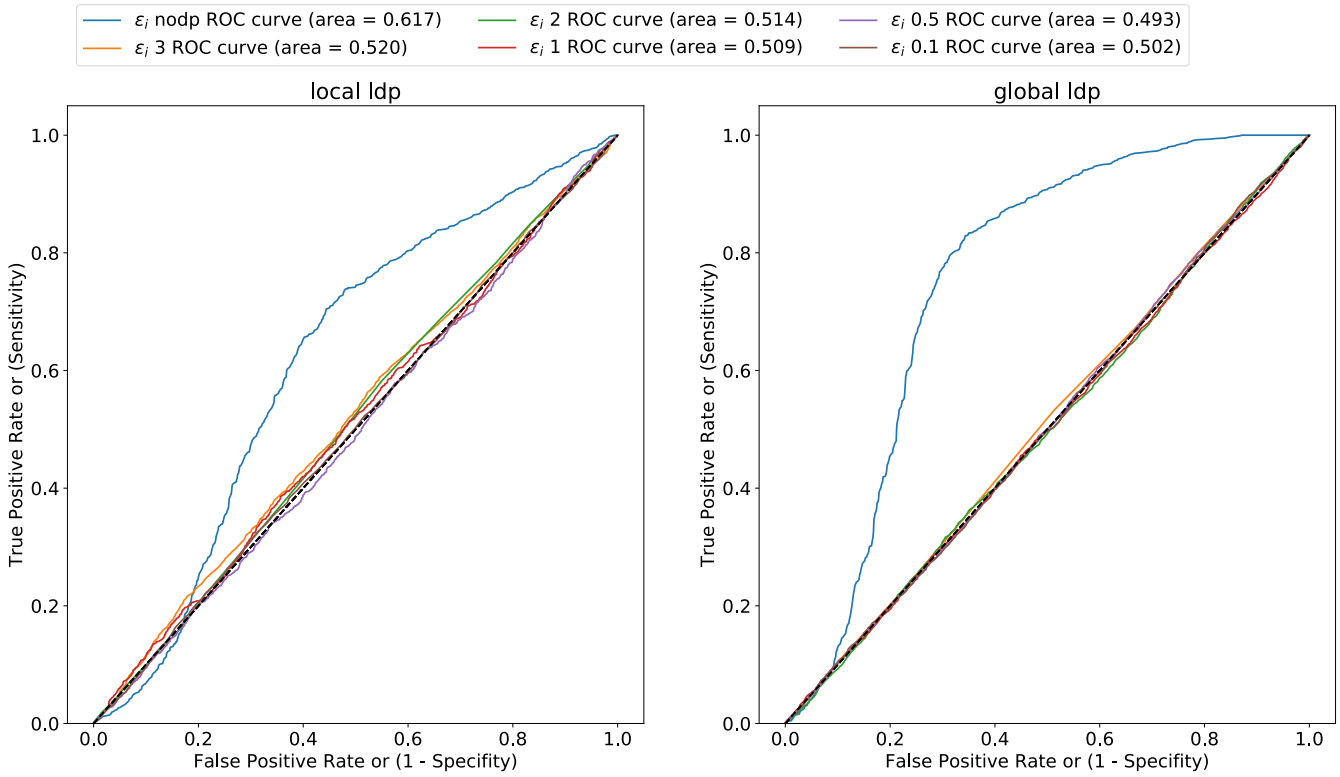
Comparison of DP learning techniques For the Texas experiments CDP and LDP yield similar defense benefits. For LDP $\epsilon_i \geq 3$ and CDP $\epsilon \geq 64.13$ the attack performance drops to baseline. Texas100 seems to be more susceptible to MIA. Which suggests the same pattern as the one observed for Purchases, that less complex DNNs are more susceptible to MIA.

ϕ **results** In this case ϕ shows best results for CDP $\epsilon \in \{64.13, 3.29\}$ depending on the class configuration. LDP, however, shows worse trade-offs compared to CDP. Only ϵ_i for $C = 300$ is favorable in the local attack. While the global attack using LDP shows that $\epsilon_i = 3$ is favorable, especially for $C = 200$. Overall it seems, that it is easier to find good trade-offs for $C \in \{200, 300\}$.

In Figure 4.14 the Receiver-Operating-Characteristic curve for the Texas100 MI attacker can be seen. Every target model using DP techniques evades the MI attack completely, because, apparently, all lines but the NoDP one, are close to the base line 0.5. The attacker ROC curves and AUC values are very similar to those from the non-federated Texas experiments from Bernau et al. [5].



(a)



(b)

Figure 4.14: ROC Curve for MIA on Texas100 dataset

4.4.3 Labeled Faces in the Wild

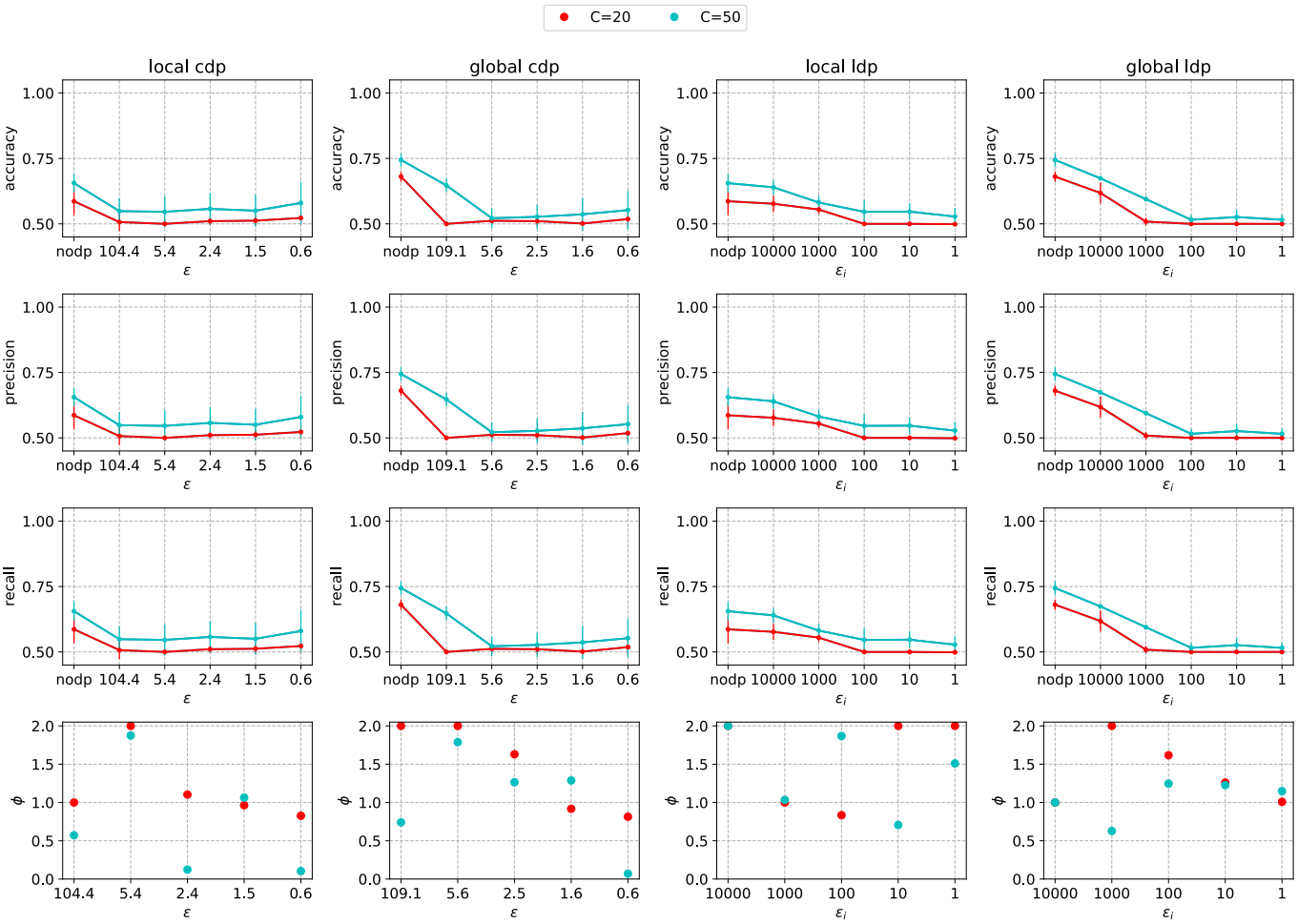


Figure 4.15: MI attack model performance on LFW dataset.

In the plots at Figure 4.15 the attack performance measured in precision, accuracy and recall for the LFW[20, 50] dataset with different DP configurations can be seen. The plot is, again, separated in four columns each containing four plots with the results of a global or local attack and either CDP or LDP DP-mechanisms. The last row measures ϕ.
Compared to the Purchases MI results, we observe, that the larger DNN configuration with 50 output neurons is more susceptible to the MI attack. Furthermore, the global attack still outperforms the local attack as already seen in the Purchases experiments. Note that the retention probabilities and Federated Learning parameters stay the same as in the Purchases experi-

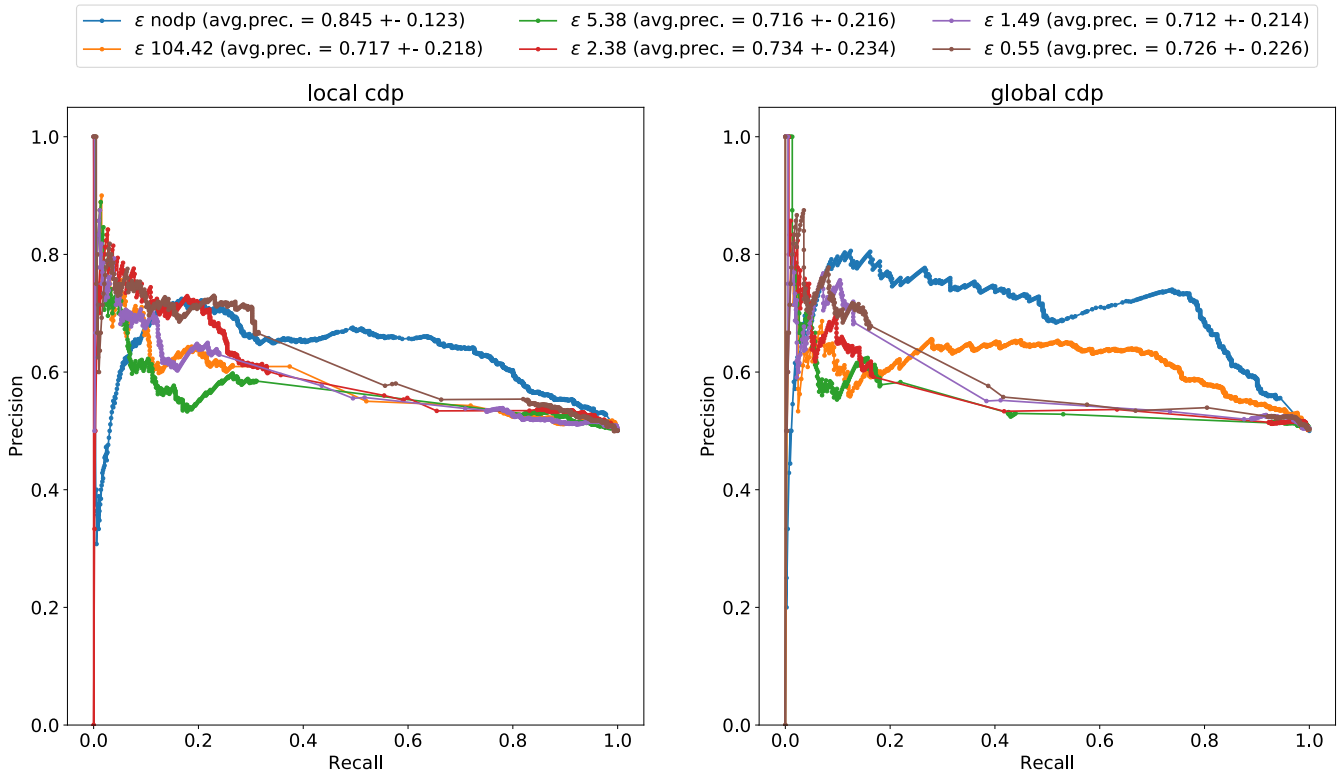
ments.

Comparison of global and local attack Here, the global attacker achieves an accuracy of nearly 75% for LFW50 and the local attacker is only at around 60% with a larger standard deviation.

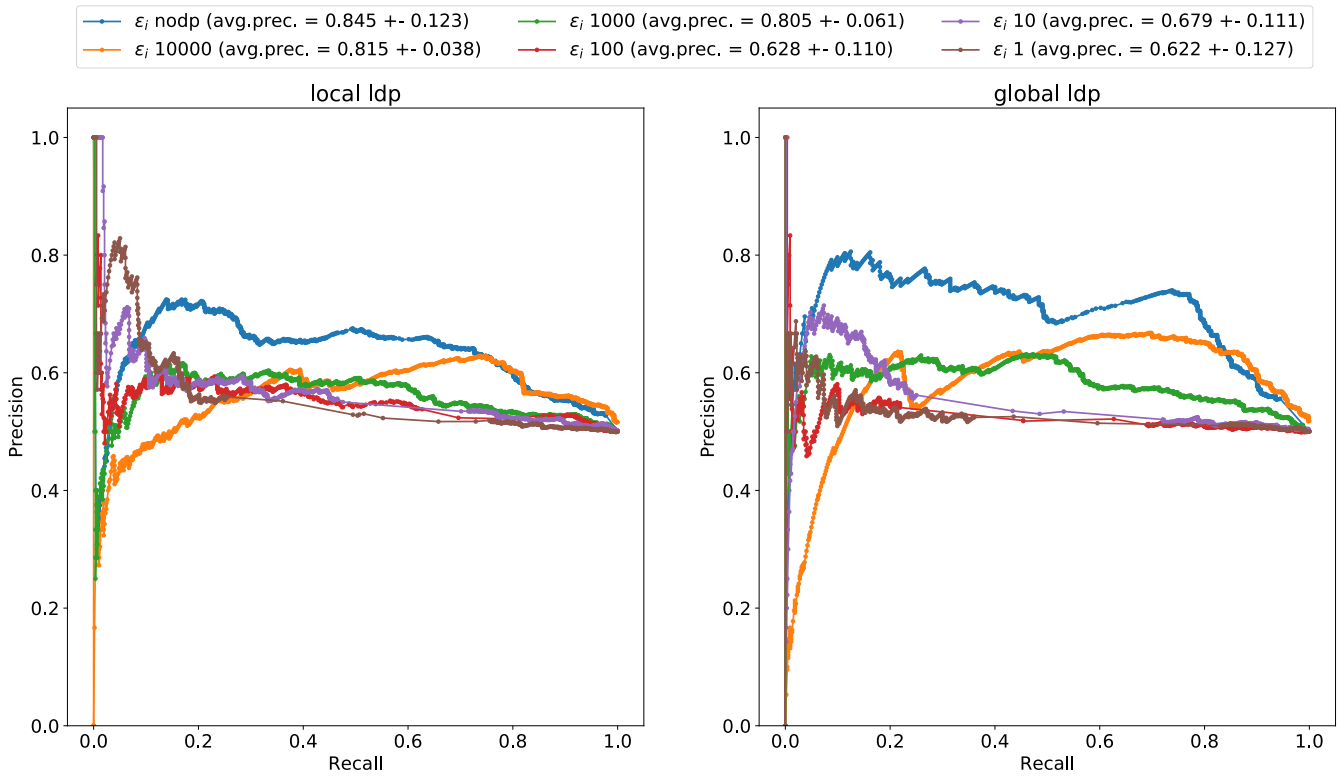
Comparison of DP learning techniques In the CDP local attack results, the accuracy on LFW50 is not reaching baseline, even with an $\epsilon = 0.55$. However, the standard deviation is very large throughout the local CDP attack accuracies. In the global CDP case, compared to purchases, the attacker still has success with a high ϵ. While in the Purchases experiments, the CDP noise brought the attack immediately to baseline 0.5. The LDP defense is again slightly weaker than the CDP mechanism. The MIA on LFW20 reaches baseline with $\epsilon_i \leq 100$ while on LFW50 LDP reaches baseline with $\epsilon \leq 1$.

ϕ **results** With CDP, ϕ gave best results with $\epsilon = 5.38$ for both target model configurations and both DP learning mechanisms. In the LDP case it is not as obvious. Approximately $\epsilon_i = 1000$ gave the best ϕ. However, the MIA showed that for this parameter it is still possible to extract some membership information.

In Figure 4.16 the precision-recall curve for LFW50 with CDP and LDP applied target models is presented, due to the LFW50 target models constituting the strongest attacker. Some attack classifier, e.g. LFW50 CDP global $\epsilon = 1.49$, are completely non-functional, as they keep predicting 0.5 resulting in a nearly straight line in the precision-recall curves. However there are thresholds, e.g. LFW50 global LDP, where $\epsilon = 104.42$ outperforms NoDP. Bernau et al. [5] show a NoDP AUC of 75 and 50 for LFW50 and LFW20 respectively for the non-federated case. With Federated Learning an attack AUC of 60 (Figure 4.18b) and 68 (Figure 4.17a) could be achieved for LFW20 and LFW50.

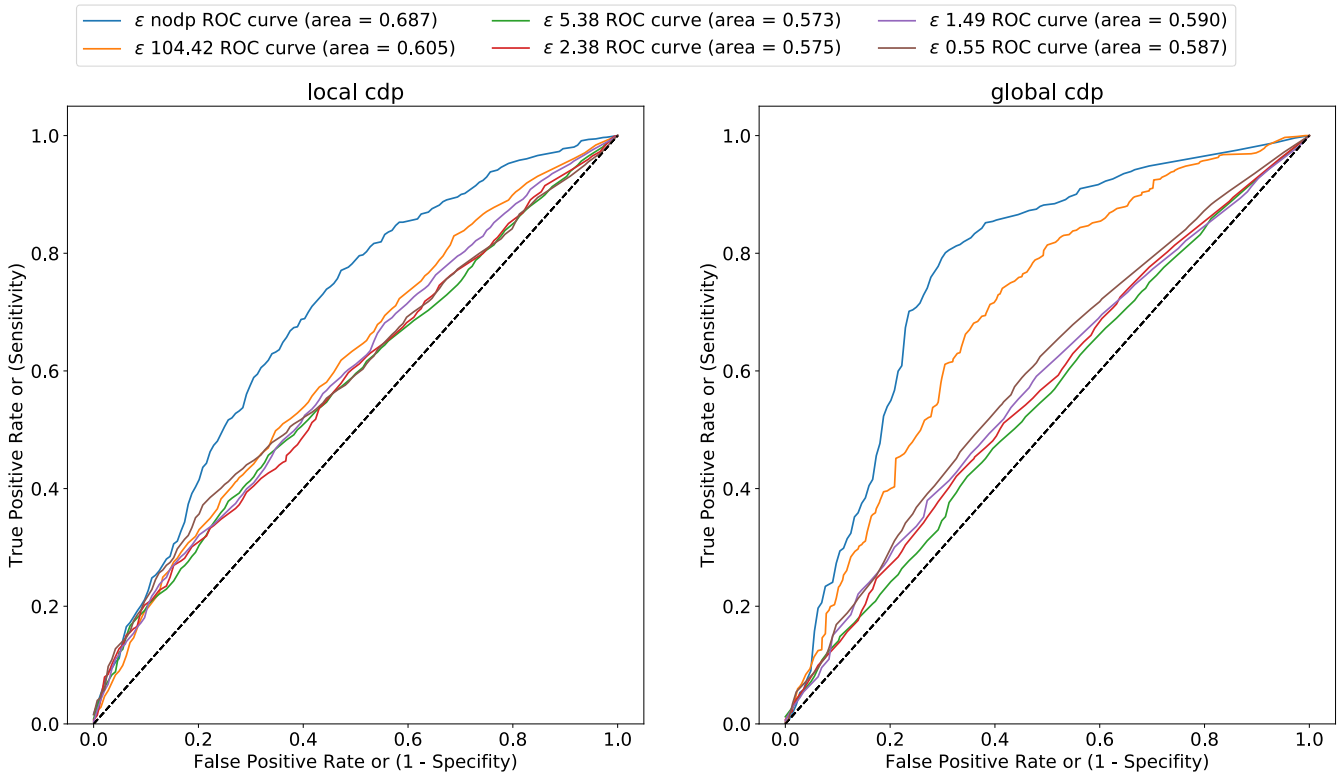


(a)

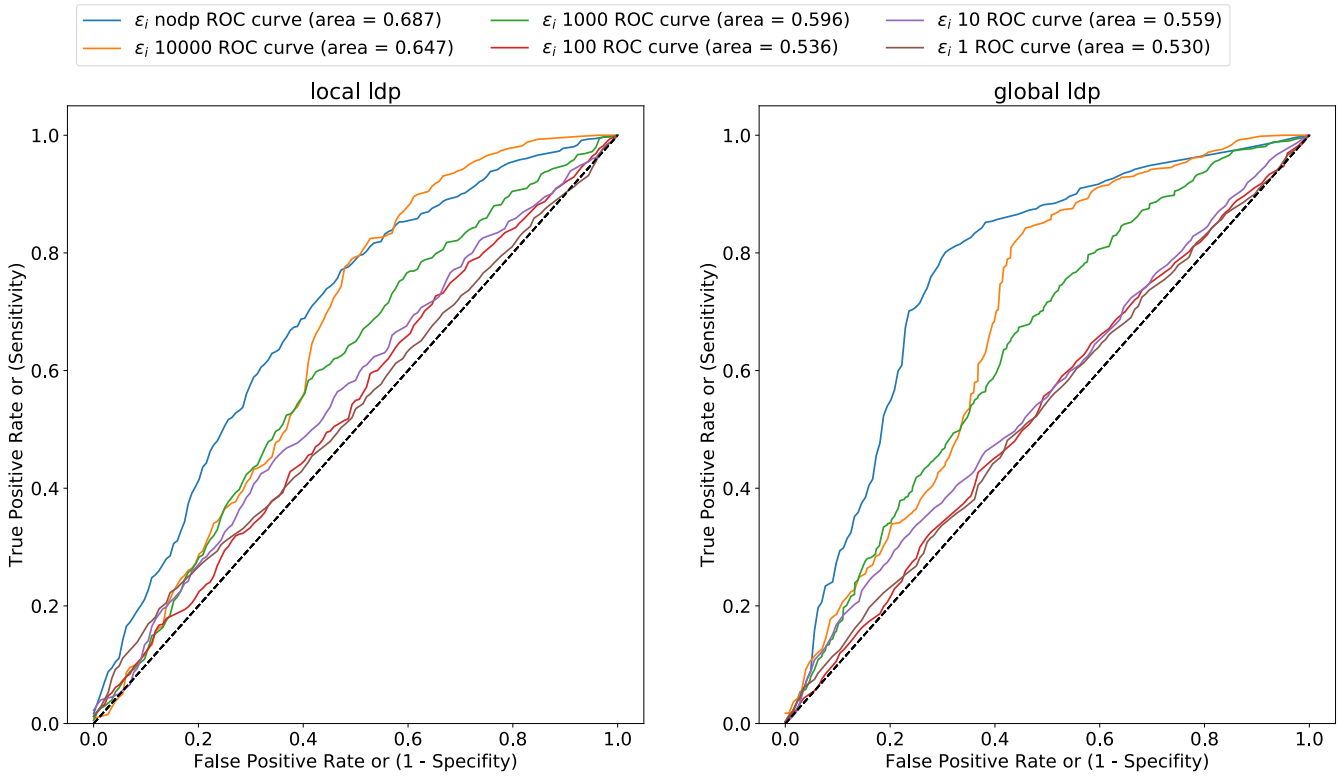


(b)

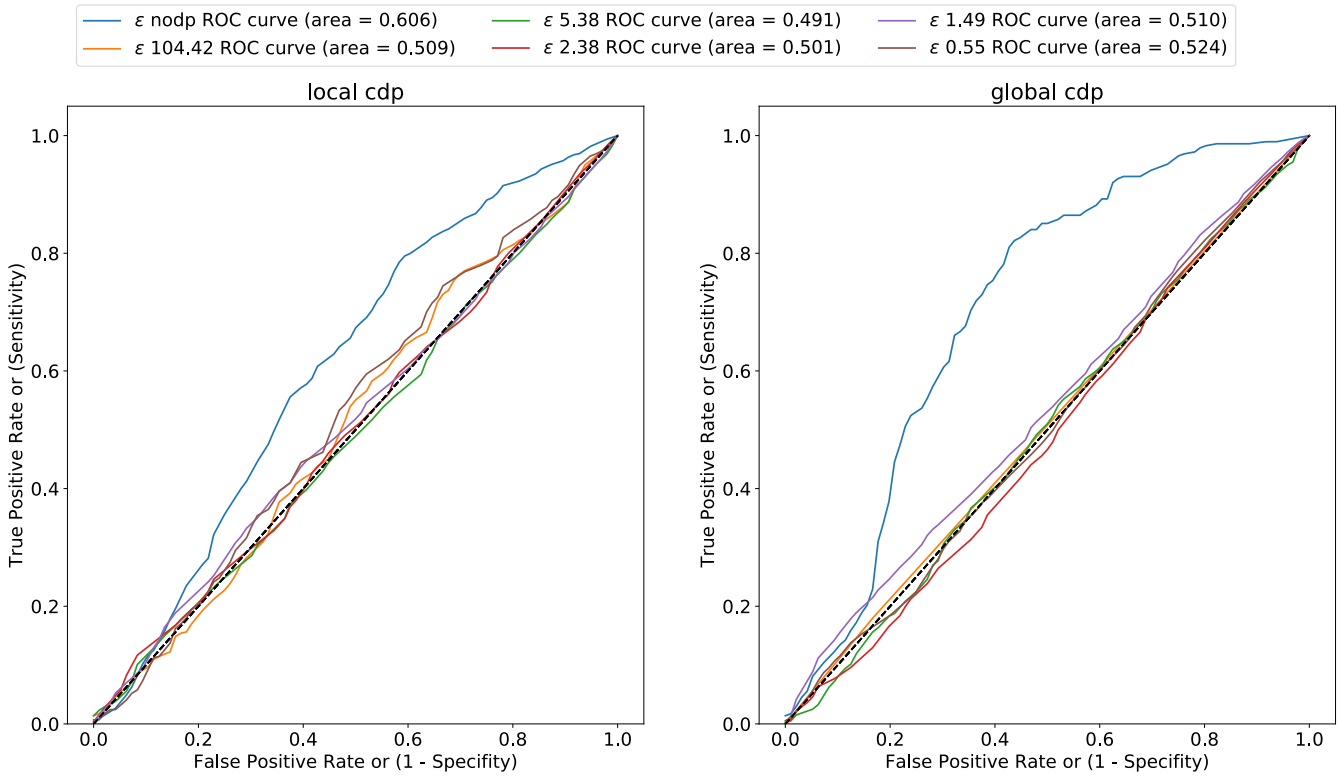
Figure 4.16: Precision-Recall Curve for MIA on LFW50 dataset



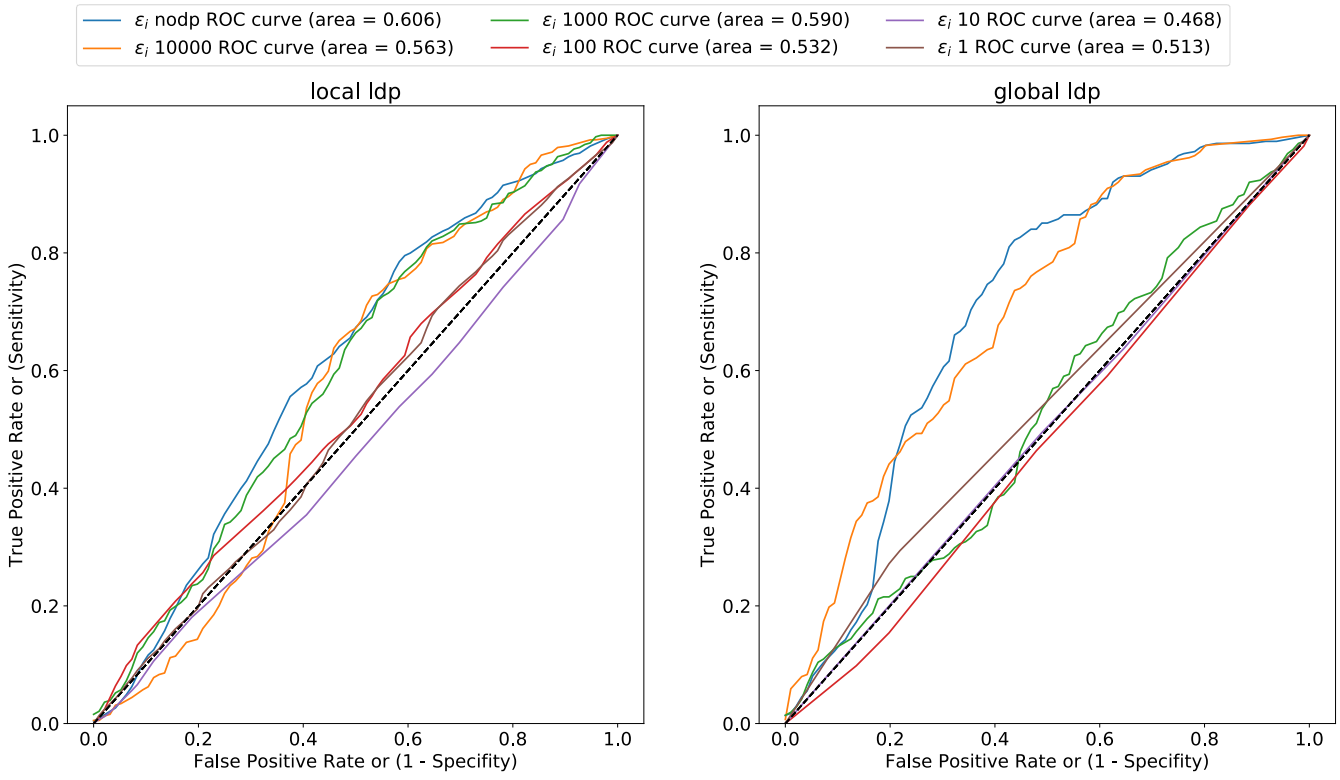
(a) ROC Curve for MIA on LFW50 dataset with CDP.



(b) ROC Curve for MIA on LFW50 dataset with LDP.



(a) ROC Curve for MIA on LFW20 dataset with CDP.



(b) ROC Curve for MIA on LFW20 dataset with LDP.

4.5 AI Attack Model Results

In this section, the Attribute Inference attack results are presented. Because the proposed attack model in this work only considers binary feature vectors, the attack model is not applicable to the LFW dataset. For each dataset, the overall attack performance measured using the attack models accuracy, precision and recall, and ϕ. The AI attack model architecture and experimental methodology is explained in Section 3.3. Retention probabilities and Federated Learning parameters stay the same.

4.5.1 Purchases Shopping Carts

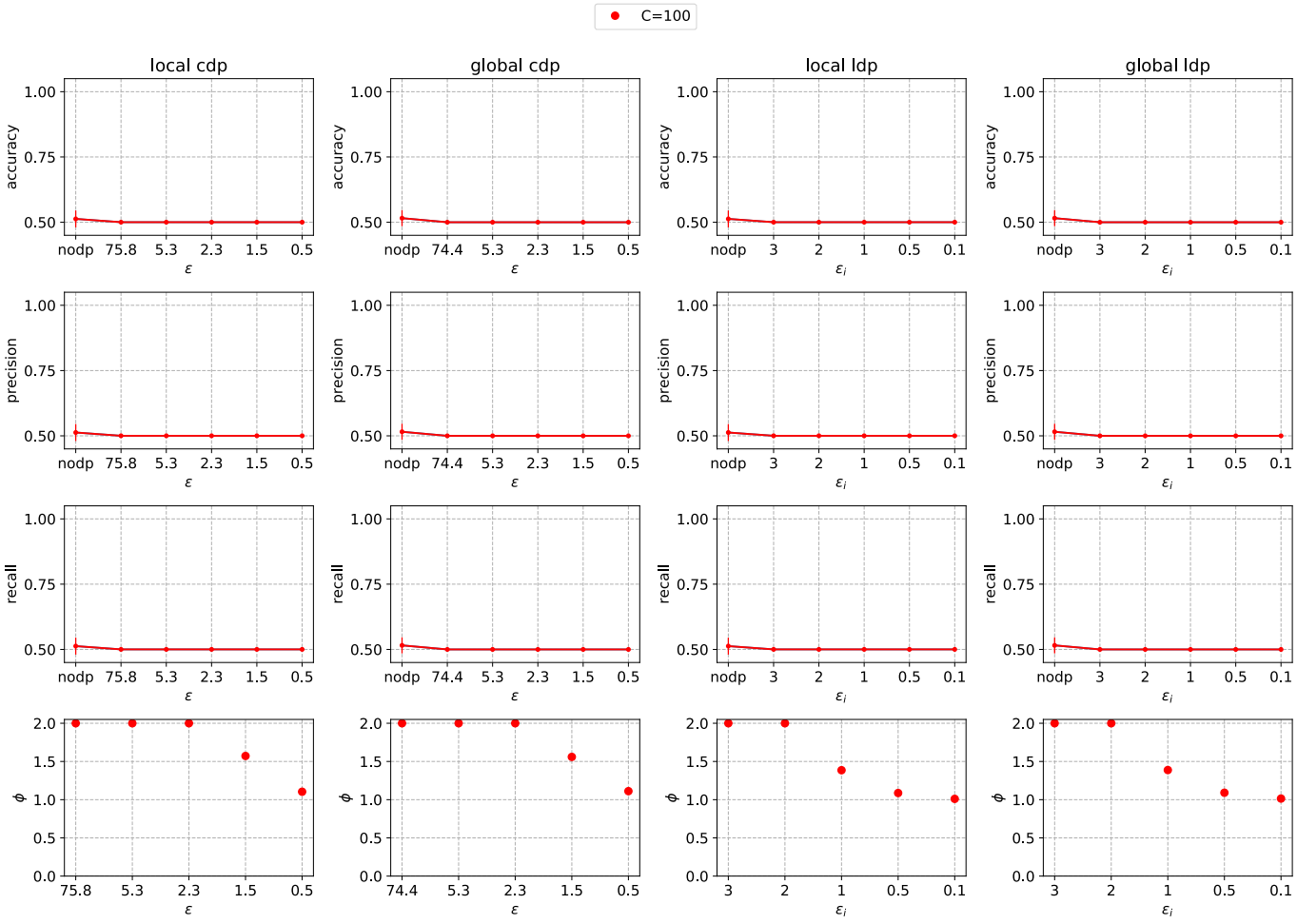


Figure 4.19: AI attack model performance on the Purchases100 dataset.

In Figure 4.19 the Attribute Inference attack model performance for the Purchases100 dataset is shown. As seen, the AI attack is not very successful and only achieves around 52% accuracy for the NoDP case. Other Purchases class configurations were not further considered, due to the poor attack performance on this particular experiment. Adding DP to the target models training, renders the attack completely non-functional. The ϕ values suggest, that $\epsilon \leq 75.84$ and $\epsilon_i \leq 3$ are sufficient as a defense against AI. This is very similar to the values received from the MI attack evaluation. Compared to the other plots the values here are rounded to only one decimal to enhance the visibility.

4.5.2 Texas Hospital Stays

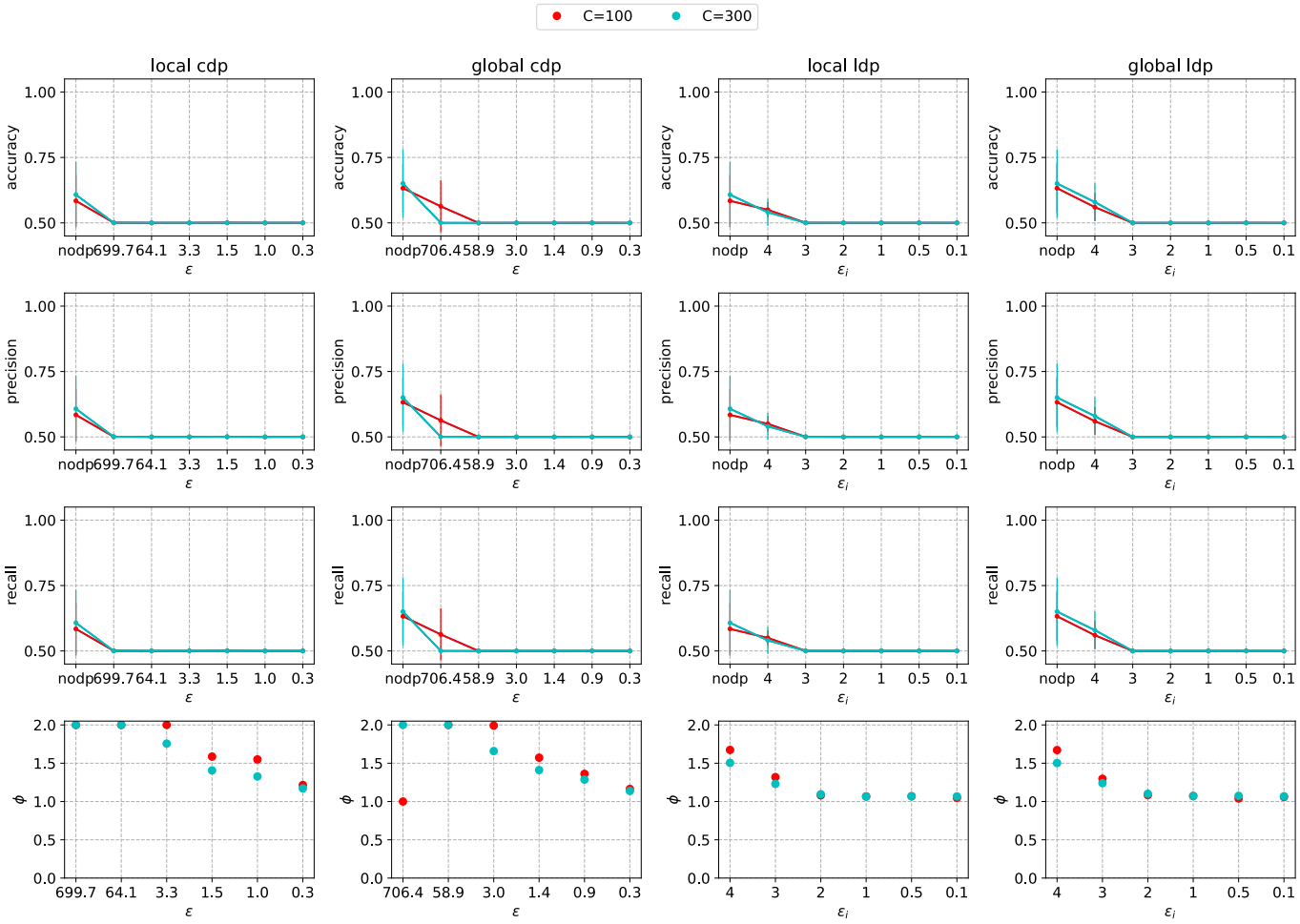


Figure 4.20: AI attack model performance on Texas dataset.

Compared to the Purchases dataset, the AI attack achieves better results on the Texas dataset. For the Texas experiments the LDP parameter $\epsilon_i = 4$, corresponding to a retention probability of 96%, and CDP $\sigma = 0.25$ yielding $\epsilon = 700$ are also evaluated, to better see for which ϵ the AIA model starts to fail. As presented in Figure 4.20, just like in the purchases dataset, the attack becomes non-functional after adding DP learning strategies to the target model. However, the local attack without DP achieves an accuracy of 55% and a global attack with an accuracy of 60%. Furthermore, the attack on Texas300 outperformed the attack on Texas100. The large standard deviation suggests, that there are some indices, that are more susceptible (75% accuracy) to Attribute Inference attacks than others (only 50% accuracy). The trade-off metric ϕ suggests an $\epsilon \geq 3.0$ and $\epsilon_i = 4$.

Discussion

This chapter is summarizing the general findings of the analysis of experimental results. Based on the analysis, the initial hypotheses are revisited and discussed. Additionally, as part of the analysis, the experimental setup and execution are critically assessed, to identify possible weaknesses and suggest potential future improvements.

5.1 Comparison of Local and Global Attack

In this work, the distributed system to train ML models is attacked both from a local and global adversary. While the global attacker has access to the local models of each data owner before they get updated, the attacker performance was expectantly better. In contrast, the local attacker only observes the aggregated global model for each communication round. The latter implicates, that the more data owner take part in the collaborative training, the worse the local attack becomes. This also has been shown by Nasr et al. [9] for the case without differential privacy.

5.2 Comparison of CDP and LDP techniques

The results support the hypothesis that there is a generally superior technique for training differentially private ML models w.r.t. the privacy-accuracy trade-off. On all datasets, CDP produces more favorable trade-offs compared to LDP. This observation is also in line with the findings of Bernau et al. [5] from non-federated learning. However, as Bernau et al. point out, LDP does not need to track the privacy account compared to CDP and has therefore a simpler application.
The ϕ metric can give an indication, whether the target model performance

decrease is more significant than the decrease in MI risk or vice versa. Although, the LDP ϵ_i are much lower than the CDP ϵ values, we can't compare them directly. ϵ_i is the privacy parameter for a single individual while ϵ is a global bound for the entire dataset and therefore a much stronger guarantee. However, the ϵ_i values can be transformed to ϵ values via:

$$\epsilon = \epsilon_i \cdot |\vec{x}|$$

With $|\vec{x}|$ being the amount of features. Table 5.1 shows the transformation of LDP ϵ_i to ϵ and compares them to CDP ϵ. This demonstrates, that LDP has in fact a much loose privacy guarantee than CDP.

Dataset	LDP ϵ_i	LDP ϵ	CDP ϵ
Purchases	$[3, 2, 1, 0.5, 0.1]$	$[1800, 1200, 300, 30]$	$[75.8, 5.3, 2.3, 1.5, 0.5]$
Texas	$[3, 2, 1, 0.5, 0.1]$	$[18000, 12000, 3000, 300]$	$[64.1, 3.3, 1.5, 1.0, 0.3]$
LFW	$[10000, 1000, 100, 10, 1]$	$[2500000, 250000, 25000, 2500, 250]$	$[104.4, 5.4, 2.4, 1.5, .0.5]$

Table 5.1: LDP ϵ_i transformed to global ϵ.

As an example, for dataset Purchases20, we have a MI precision of 62% using LDP $\epsilon = 18000$ but a base line precision of 50% using CDP $\epsilon = 75.8$.

5.3 Membership Inference Attack

This thesis uses the white-box MIA model from Nasr et al. [9]. Furthermore, this thesis shows that ϵ-DP is a strong defense against this attack model. This behavior can effectively be used to assess the anonymization in a dataset. In a federated setting however, the local attack introduced by Nasr et al. gets worse, indicating that for a fair amount of data owner, no data owner can infer sensitive membership information about other data owners. Although, the global attack would just work fine, making it the essential trust anchor in any FL system. During the work on this thesis, another phenomenon could be observed: The attack model achieved a much higher accuracy when using stratified sampling according to each dataset label. This was not always possible, because some datasets only have a low representation for certain classes and it turned out, that some data owner lack some labels. This means a stratified sampling per data owner was not possible. However, the insight indicates that there are some classes that leak more membership information than other.
In general, using this white-box Membership Inference attack to measure

the needed privacy guaranty is not trivial: MIA attack features have to be generated, the model has to be fine-tuned using a costly hyperparameter search and lastly it needs to be trained and evaluated with suitable metrics. Although, using a probabilistic hyperparameter search instead of e.g. a grid-search, simplifies this process. More optimization techniques like dimension reduction of the gradients using e.g. SVD or PCA could be tested.
The white-box MI model from Nasr et al. has been significantly improved. Introducing a convolution *width* parameter and replacing the *ReLu* layer with a *LeakyReLu* stabilizes the training and yields better accuracy. Furthermore the model selection process differs from Nasr et al. Instead of taking the best model out of 100, we use *early-stopping* and a probabilistic hyperparameter search to find suiting parameters for every particular dataset.

5.4 Attribute Inference Attack

The Attribute Inference attack model in this thesis is novel and based on Nasr et al. [9] Membership Inference attack model. In most evaluations it outperformed the AIA introduced by Yeom et al. [16]. Furthermore, this work shows that MIA is a generalization of AIA. However, while the AIA performances for Texas and Purchases are not very significant, they show that there are certain indices that can be inferred more easily (80% precision) than others (only 50% precision). Zi et al. [43] experimentally demonstrate, that datasets with a large distance, i.e. the maximum distance for each entry to any other, perform better than datasets with small distance. They observe this phenomenon using the Purchases dataset and by measuring the Hamming distance. They remove every close entry in the Purchases dataset to achieve a certain large distance. With the larger distance, the Attribute Inference attack gets better. Intuitively this makes sense: lets consider the "MovieDB" dataset, where each binary feature represents whether an individual watched a certain movie or not. It should be relatively easy for a human to guess e.g. the individual's gender just by looking at his seen movies. However, if each individual in the dataset watches the same movies as everyone, resulting in a very low distance, it is nearly impossible to predict the gender for a single individual. This can be easily extended to scalar datasets using another distance metric like the Manhattan- or Euclidean-distance.
In this work, another measurement has been developed to measure how effective AIA will be on a dataset. Instead of taking the Hamming distance of each individual, we take the fraction $dist(\mathcal{D}) = \frac{n}{|\mathcal{D}_u|}$ on the overall size of the dataset n over the amount of all unique entries $|\mathcal{D}_u|$. A $dist(\mathcal{D}) = 1$ corresponds to a Hamming distance of 0 for a binary dataset like Purchases

or Texas. For some datasets we have following results (seen in Table 5.2):

Dataset	$dist(\mathcal{D})$
Purchases10	1
Purchases100	1.00001
Texas100	1.003
Texas200	1.584
Texas300	1.584

Table 5.2: Distance measurement for different datasets.

This is in line with our experimental results, showing that Texas300 AIA yields a much better attack accuracy than the AIA on the Purchases dataset. Our distance values are not surprising, since Texas has about 6000 attributes while a single entry in the Purchases dataset has only about 600 attributes, resulting in a large space of possible feature permutations in the Texas dataset and a smaller one for the Purchases dataset.
More formally: given a binary dataset with $\vec{x} \in \mathcal{D}$ like Purchases or Texas and, moreover, given a correct sensitive attribute s and the same attribute flipped resulting in a wrong attribute $\hat{s}$, we'll have a joint probability distribution of $P(\vec{x}, s)$ and a joint probability distribution with $P(\vec{x}, \hat{s})$. We can then determine the Posterior probabilities $P(s|\vec{x})$ and $P(\hat{s}|\vec{x})$. With a distance of 1 both posterior probabilities must have a similar probability (0.5) and there is no **correct** or **incorrect** sensitive attribute that could be inferred from the ground truth. However, if the posterior probability of $P(s|\vec{x})$ is large and consequently the posterior probability of $P(\hat{s}|\vec{x})$ low, the distance converges towards n. The posterior probability gives a strong bound on how effective an AIA model could be. In theory, a dataset with a distance close to 1 should need a lower privacy guarantee (larger ϵ) and a dataset with a large distance should need a stronger privacy guarantee (lower ϵ) to defend against AIAs.
In this thesis, using AIA as reference for assessing privacy parameter ϵ seems to be inferior for using MIA. Not just because it performed way worse, but the experiments demand a lot more computational resources, due to the fact that for each attribute one MIA model is needed.
LDP perturbs single entries in a dataset compared to CDP which perturbs a global aggregation function. Therefore, it would make sense to assume, that LDP yields a better defense against Attribute Inference attacks. The experiments show, that the standard deviation of the attack precision for the LDP perturbed target models are much smaller than the standard deviation of the CDP perturbed target models. This means, that the LDP perturbed

attributes can be inferred more or less equally, while in the CDP case, some attributes achieve a very high attack precision.
Even though, the experiments suggest, that still CDP yields a better privacy-utility trade-off than LDP against AIAs just like they do against MIAs.

Conclusion and Outlook

This thesis has analyzed the privacy-utility trade-off of DP learning techniques LDP and CDP for the application on ML models in Federated Learning for several datasets using two different inference attacks. During this work, a framework for the collaborative training among several data owners on a single machine using multi-processing has been developed and was shown to be successful to simulate real world Federated Learning application. This framework can be used in future works to analyze distributed ML applications. Furthermore, it has been demonstrated, that CDP and LDP do not allow to compare the techniques solely based on the chosen privacy parameters ϵ_i and ϵ. Therefore, inference attacks were used to empirically measure privacy with either of the two DP learning techniques. In this work, it is shown that, even though the data owner don't share their data, the MI and AI attacks can be very successful. In Federated Learning, there are two possible adversaries, the local and the global attacker. The global attack constitutes a stronger attack and gives therefore a stronger bound on the possibility of inferring sensitive membership information. Consequently, a stronger attack, yields a stronger privacy assumption as defense. Our experiments demonstrated, that the more data owners participate in the training the lower the risk of a local attack. Although the risk of a global attack remains constant. Traditionally, LDP is preferred, when a curator (i.e. the server which computes a ML model) such as found in cloud based machine learning service platforms that are potentially untrustworthy. Uploading noisy data poses a lower privacy risk. In a Federated Learning setting, however, no one but the actual data owner sees the potentially sensitive data.
In this thesis, a novel Attribute Inference attack is introduced. Although the attack outperformed some state-of-the-art attacks, its performance was lower than the Membership Inference attacks. Empirically, we demonstrated, that although LDP perturbs local data and should therefore yield a better

anonymization, it was still equal or inferior to the CDP technique. CDP, whatsoever, needs a completely new hyperparameter search for the ML model and moreover needs its privacy accountant to be tracked during the training. Although, using RDP accountant, CDP has a much larger privacy bound compared to LDP for the same loss in attack accuracy, which makes it superior.

The white-box MI attack implemented and analyzed in this work has the potential to serve as benchmark for empirically analyzing the worst-case MI risk on federated trained ML models. The original white-box model has been modified and optimized to further yield a stronger attack performance. Assessing the privacy loss in a model with MIA is only suitable when the model is optimized. This poses an arguably non-trivial and complex task for real world application.

We empirically demonstrated, that the Attribute Inference attack performance strongly depends on the distance of a dataset. In a future work, the mathematical connection of the distance and the dependence on the ϵ parameter could be formally proven and bounded.

Another differential privacy technique called PATE [15] could be analyzed with the very same methodology from this thesis in a future work. In PATE multiple data owners train on a disjoint subset of data just like in this work's experiments. The prediction is generated using a voting over the data owner model ensemble instead of using a global model. ϵ-DP is achieved by perturbing the votes using a Gaussian mechanism. It could be analyzed, whether the PATE yields a better general utility-privacy-trade-off than the methods described in this work. Furthermore, the application of LDP in PATE could be assessed, for instance, the participants in PATE would have to train on a perturbed dataset and the voting system should be unperturbed.

Nasr et al. [9] suggest extending the white-box MI architecture to an auto-encoder DNN architecture to be able to train the DNN unsupervised. Whatsoever, we suspect a stronger latent space encoding using the unsupervised training. And then for fine-tuning a supervised training on the already optimized encoder. This could yield an even better accuracy for MI and AI attacks.

Lastly, in reality, the datasets of each data owner are not only disjoint, but also very different in terms of labels. In an experiment the *MNIST* dataset was trained using Federated Learning, where one data owner trained on the class labels $[0, 1, 2, 3, 4]$ and the other one training on $[5, 6, 7, 8, 9]$. The MI attack was considerably worse than using just a disjoint data distribution for the target models. In the future, more research into the analysis of different data distributions in Federated Learning and their consequence in privacy guarantees could be done.

Appendices

Parameter Evaluations

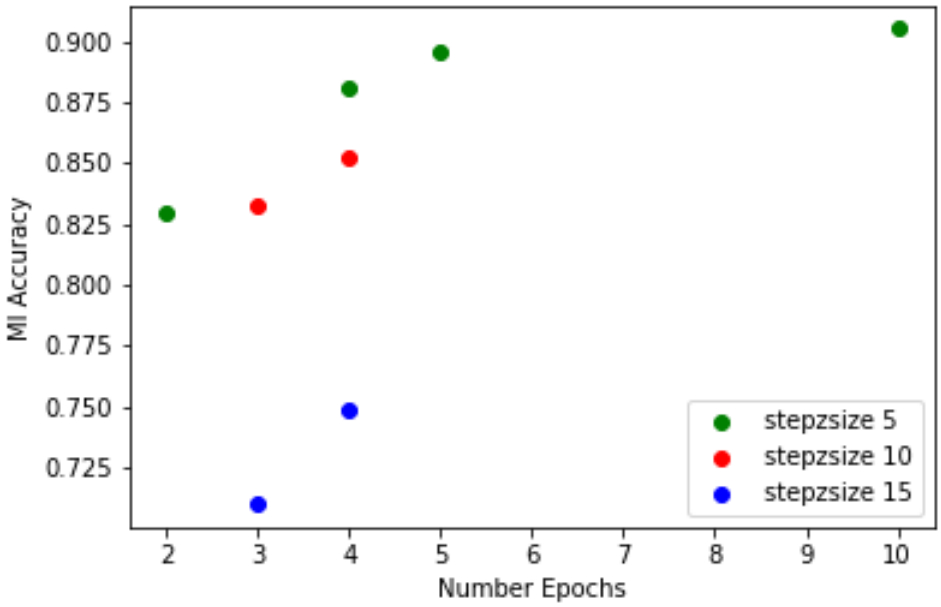


Figure A.1: MI performance using different T.

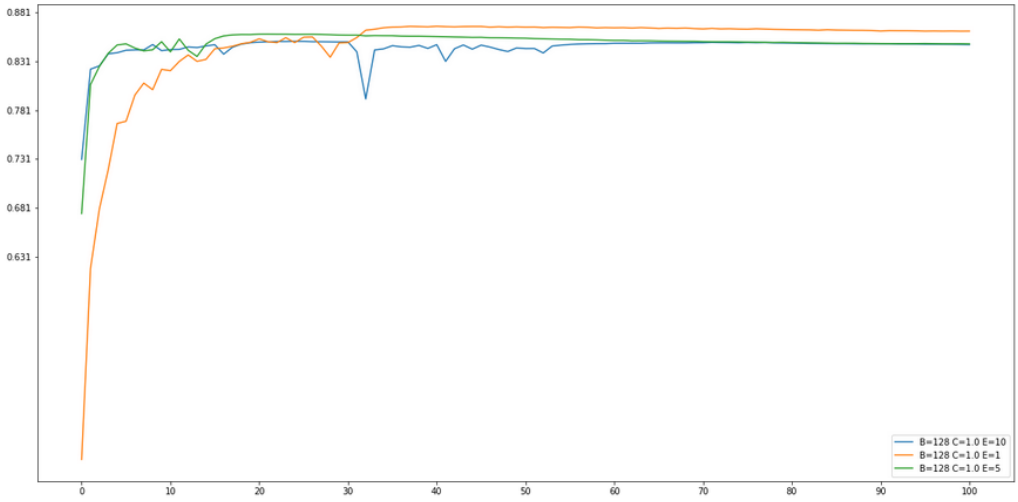


Figure A.2: Target model training with different E.

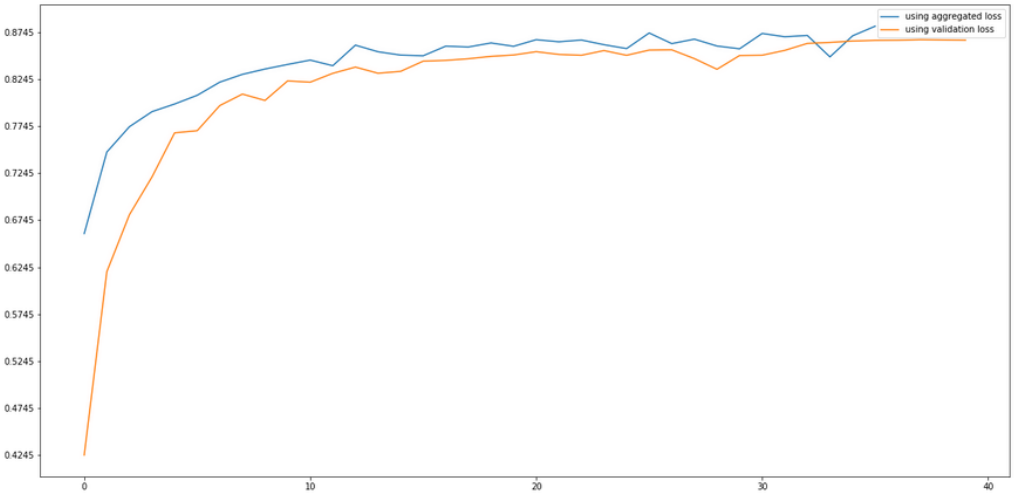
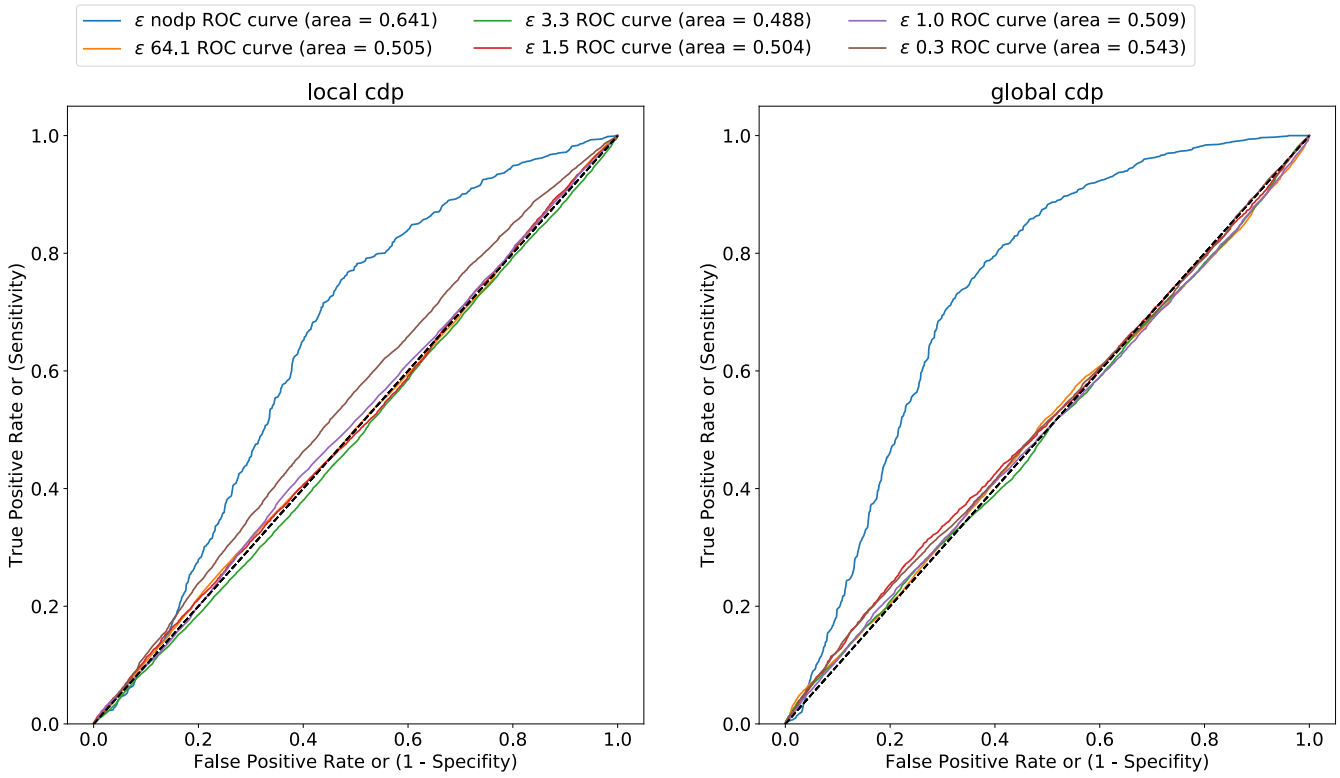
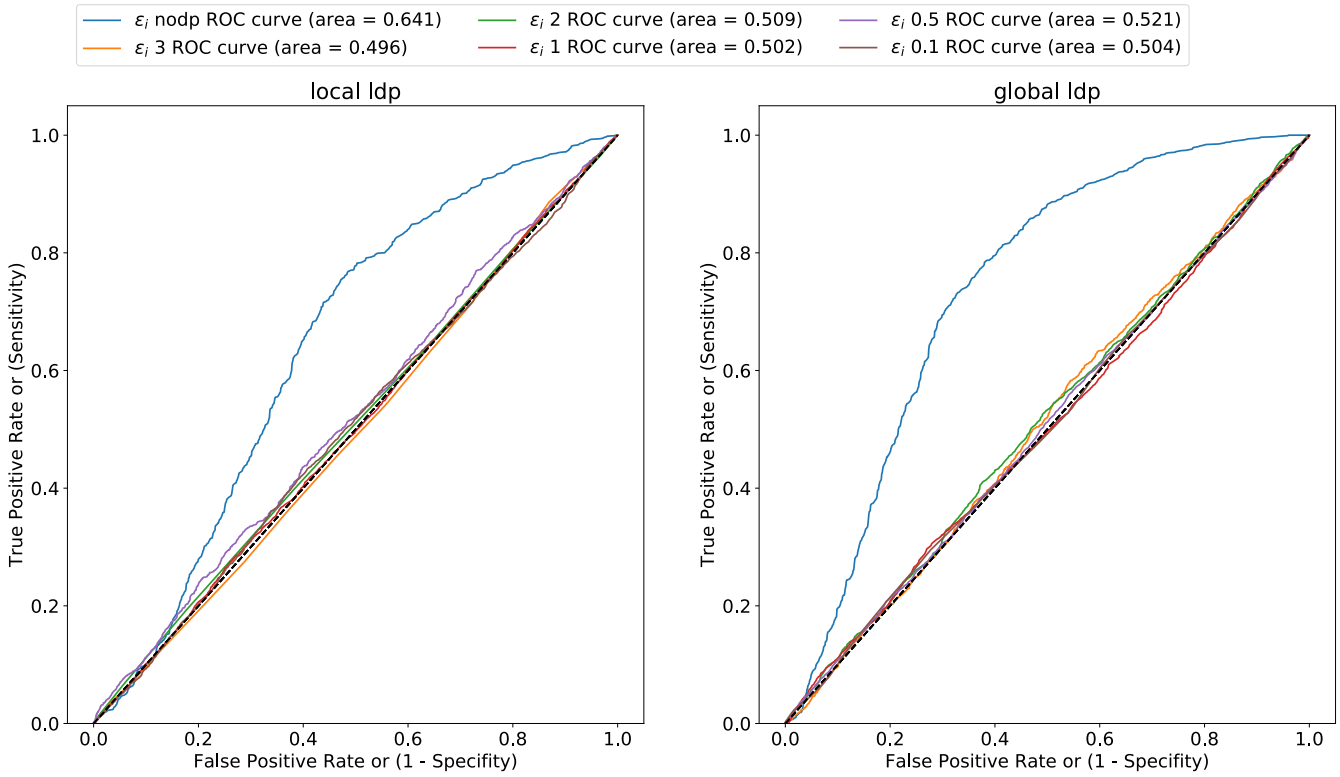


Figure A.3: Different loss measurements compared.

Attack Model Evaluations

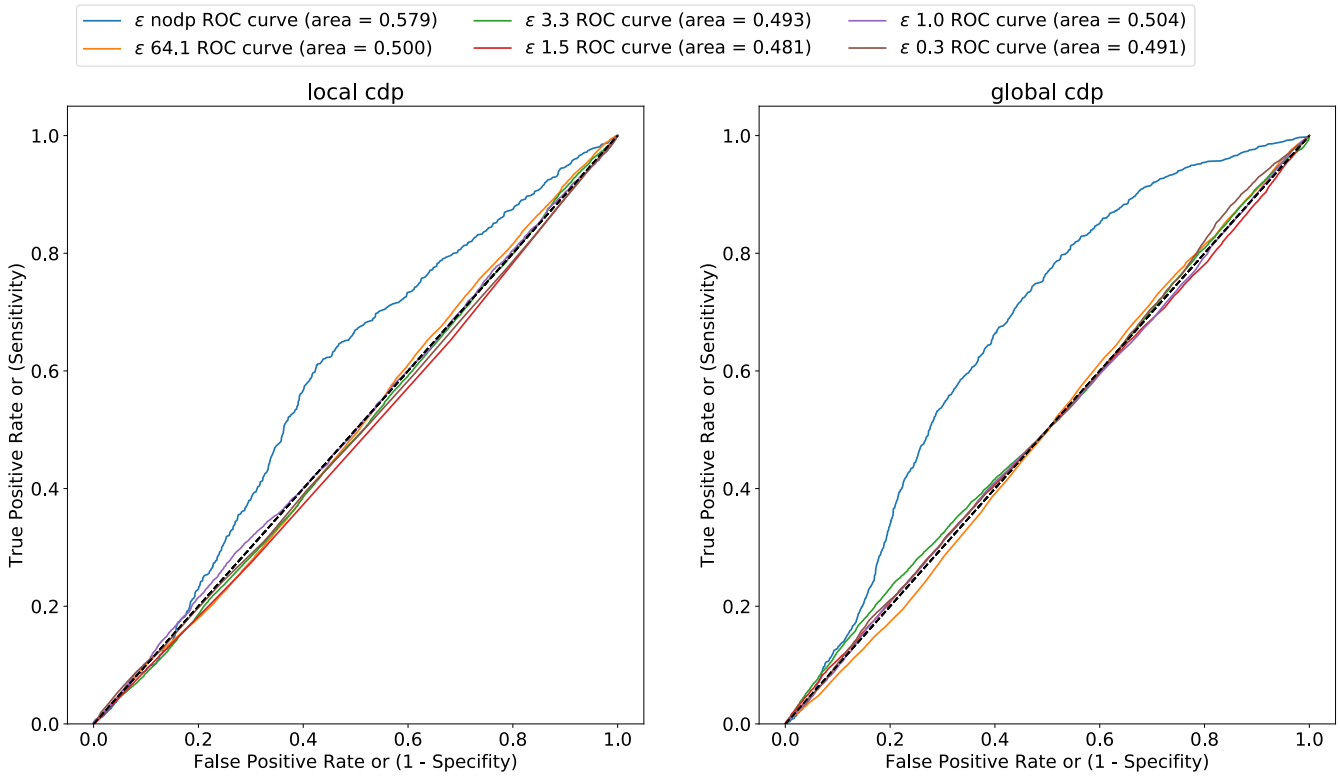


(a)

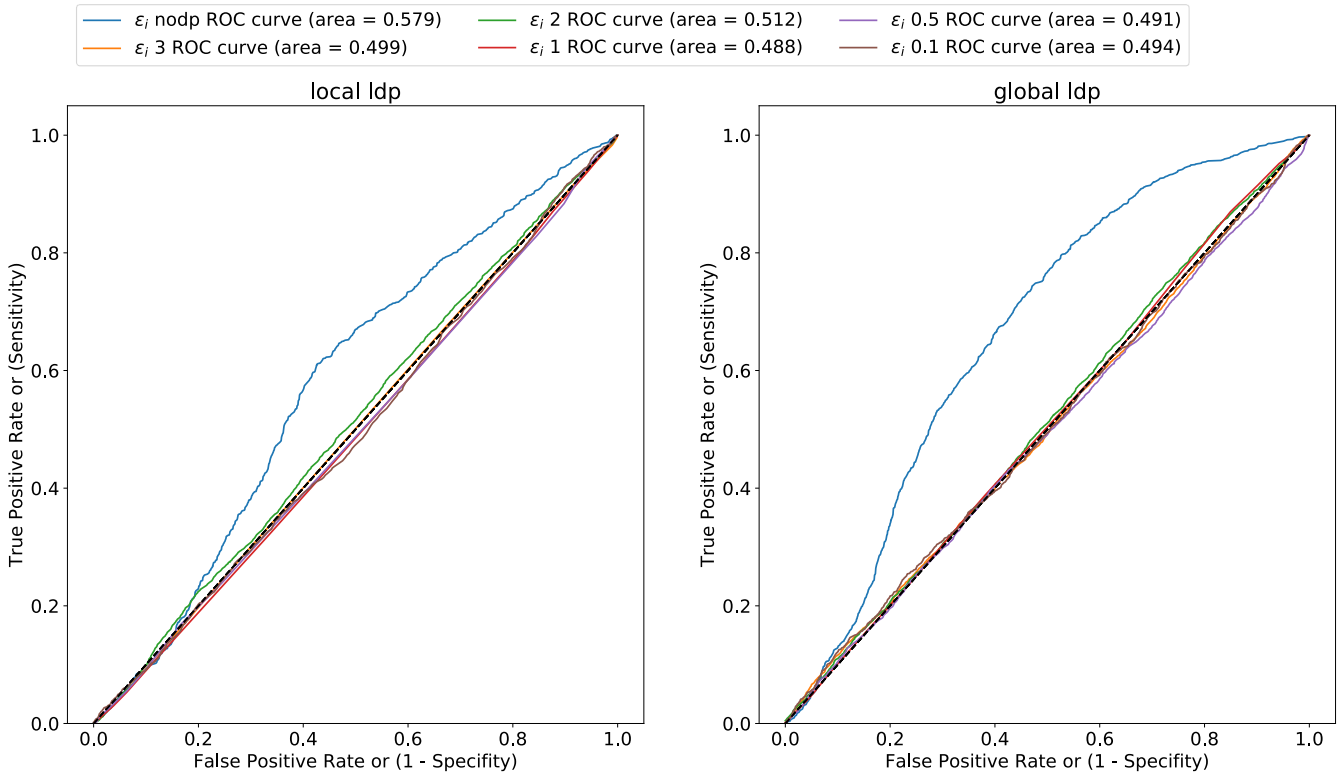


(b)

Figure B.1: ROC Curve for MIA on Texas150 dataset.

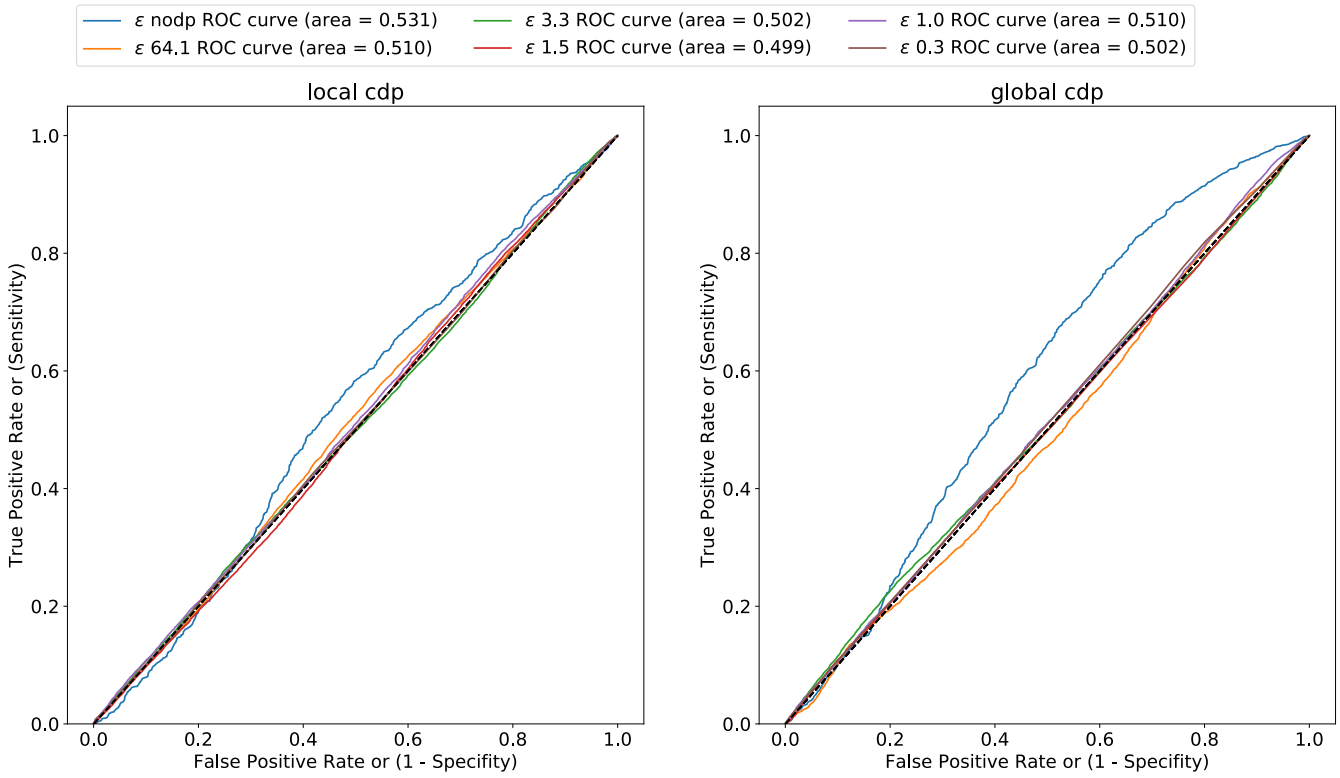


(a)

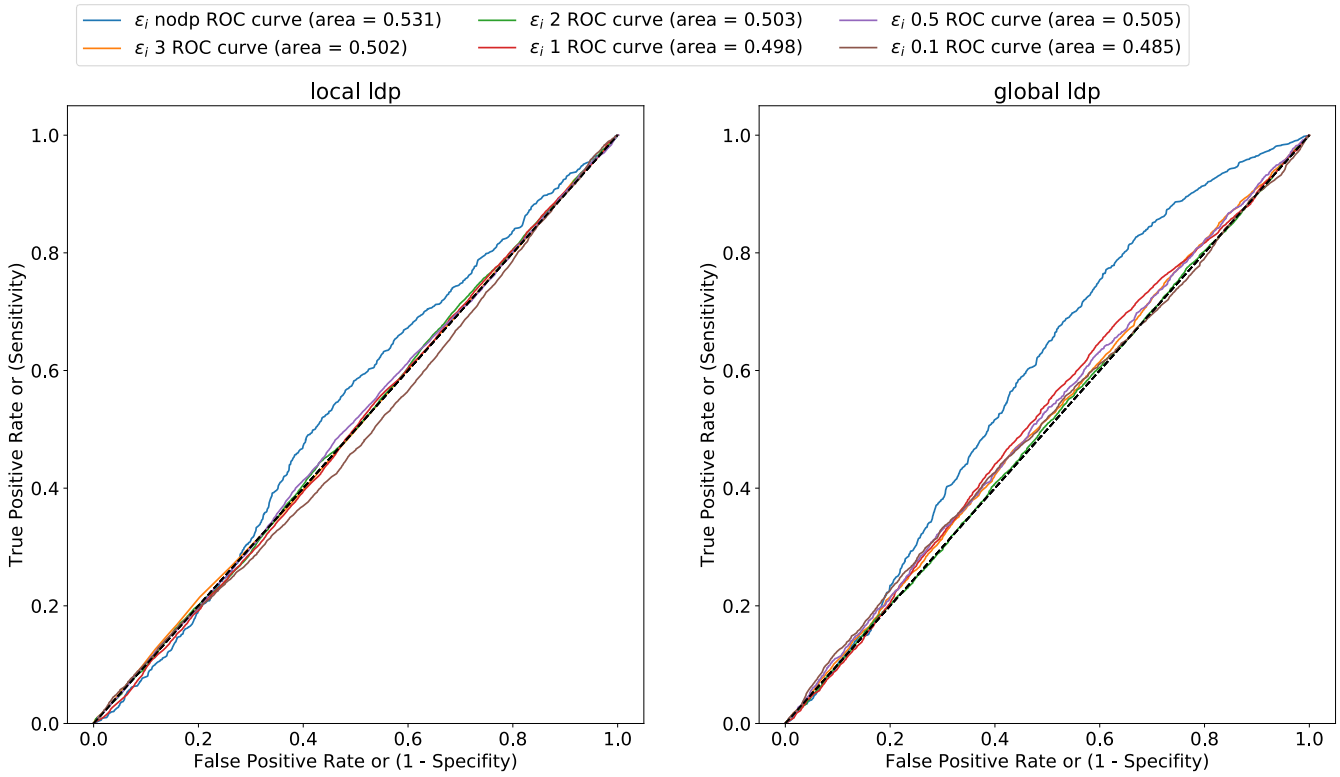


(b)

Figure B.2: ROC Curve for MIA on Texas200 dataset.

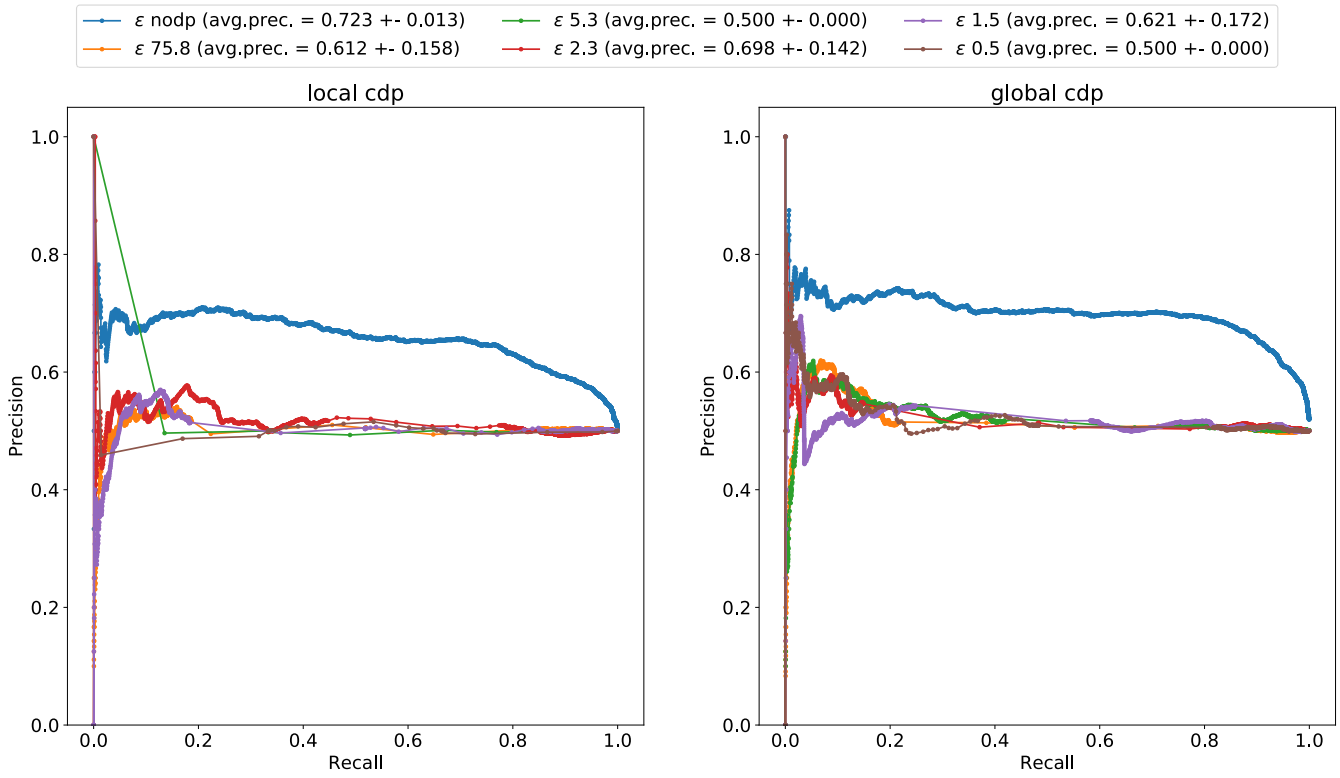


(a)

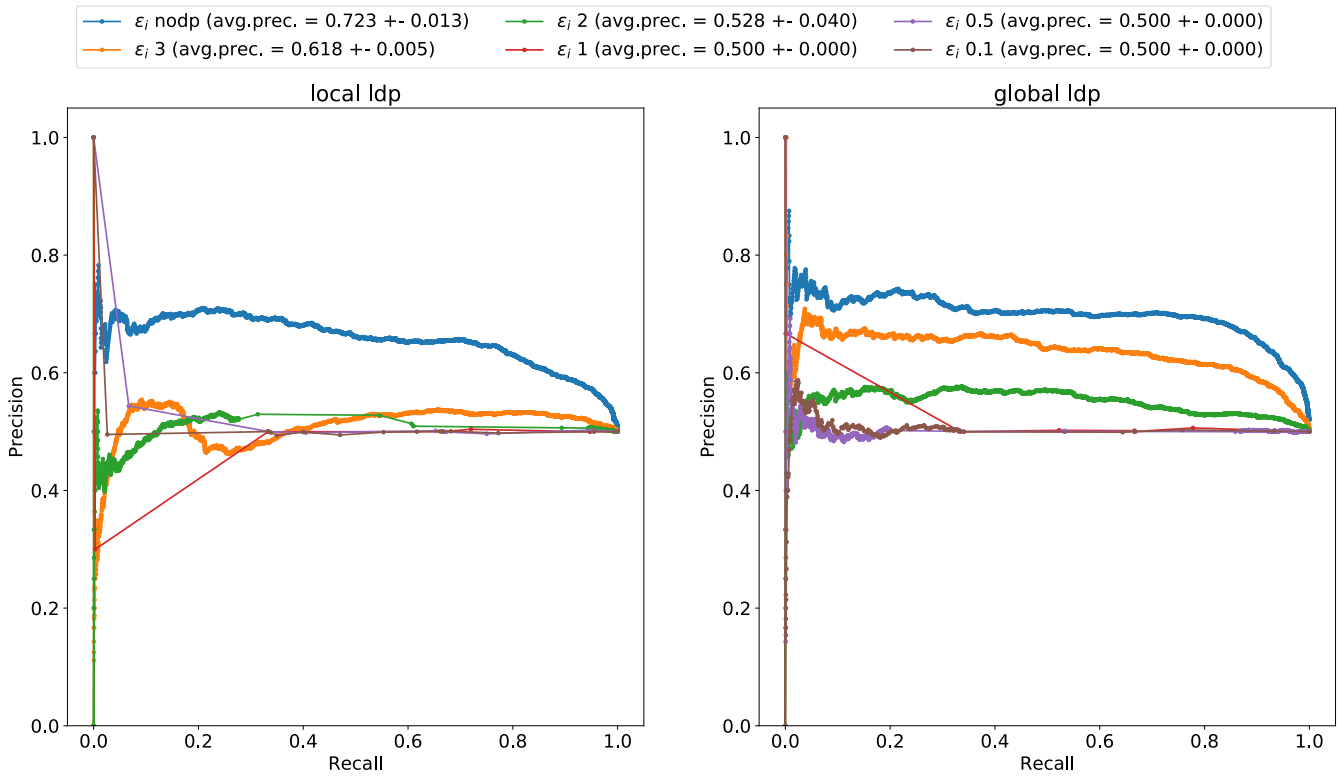


(b)

Figure B.3: ROC Curve for MIA on Texas300 dataset.

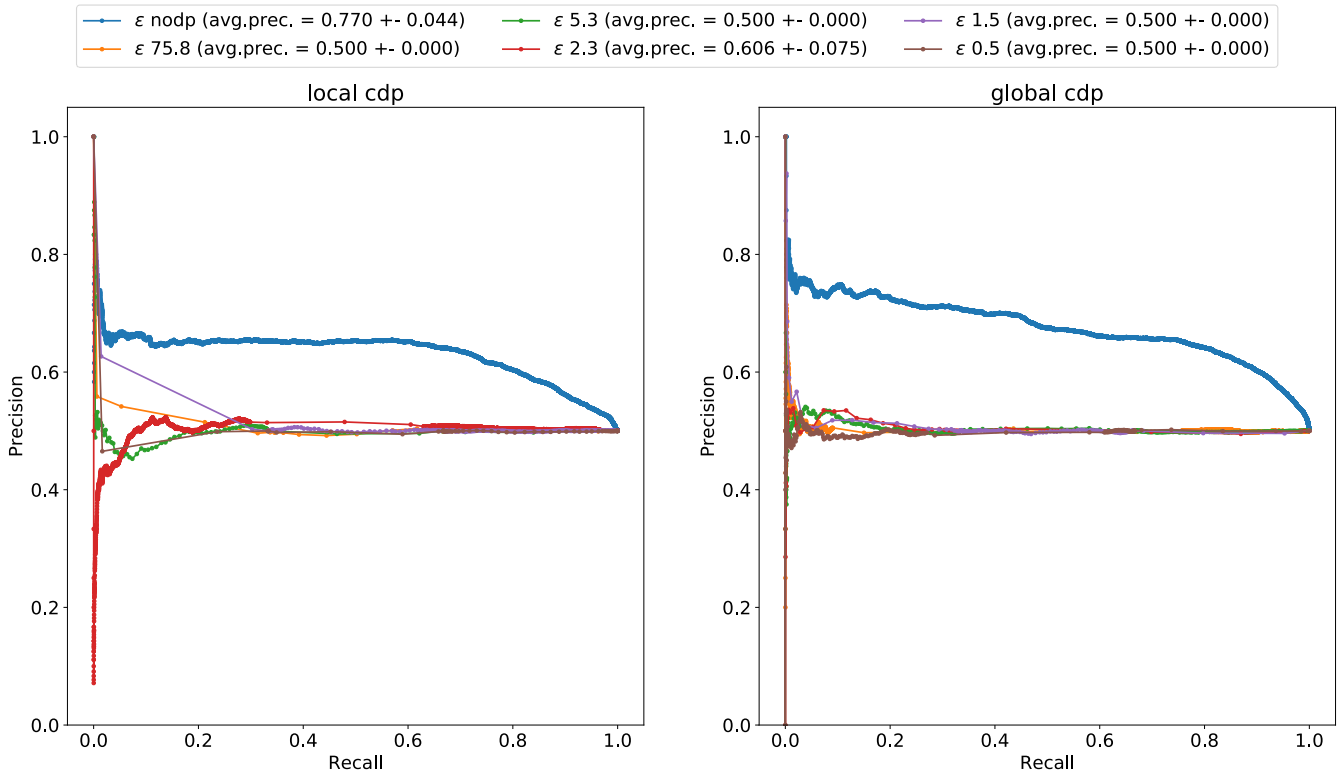


(a)

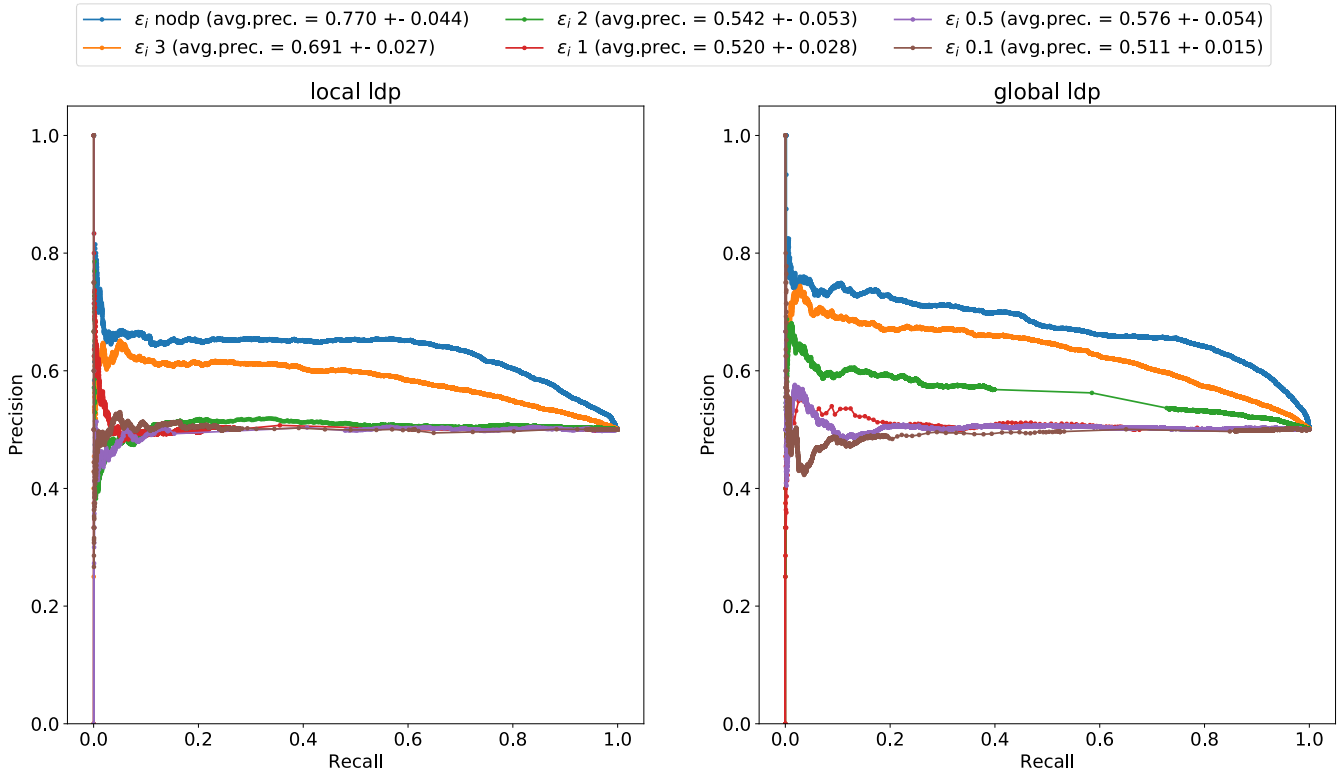


(b)

Figure B.4: Precision-Recall Curve for MIA on Purchases10 dataset.

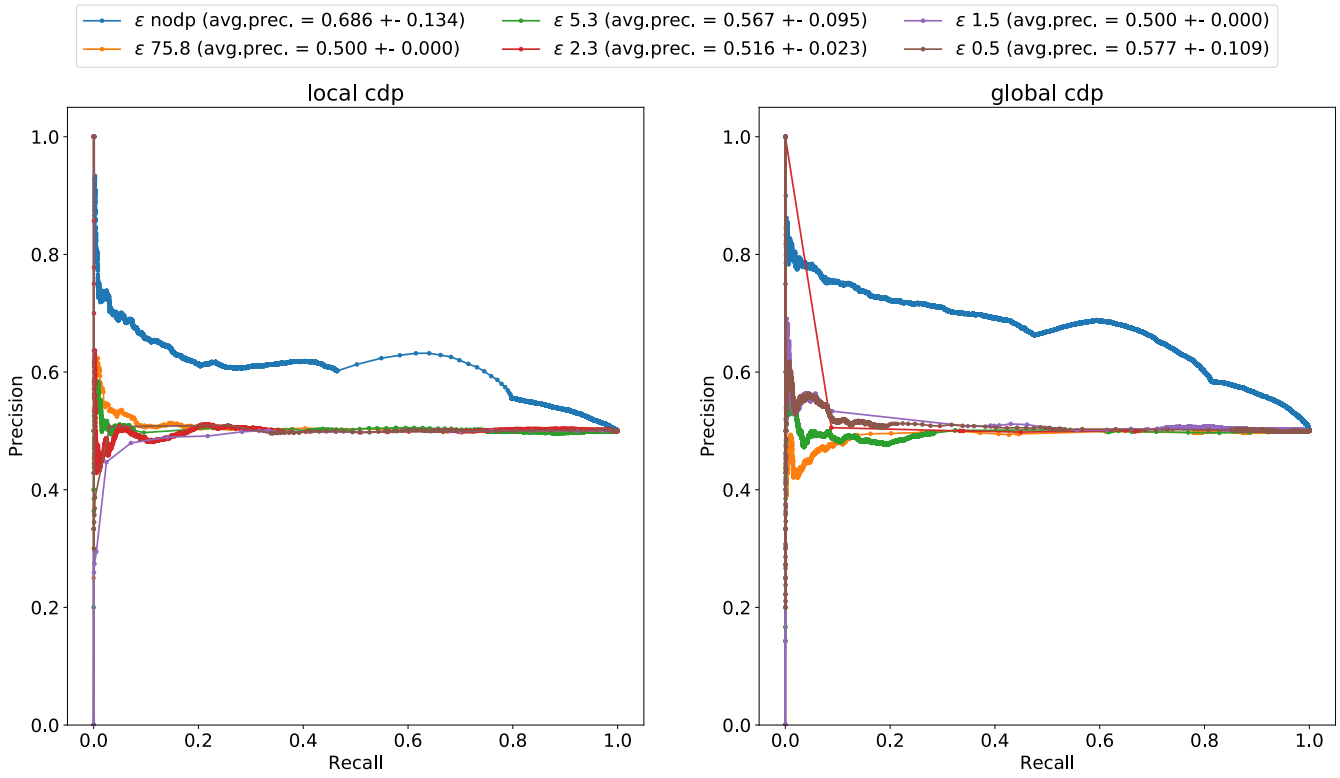


(a)

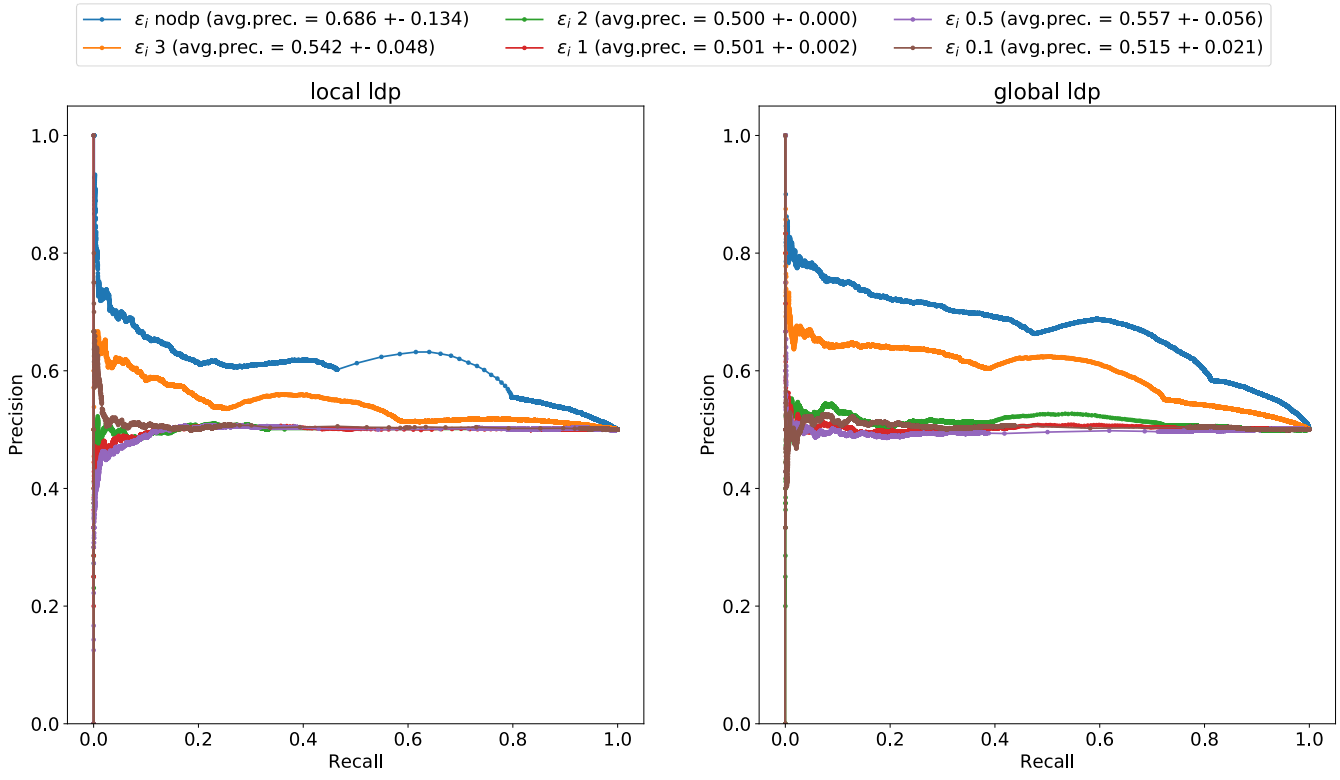


(b)

Figure B.5: Precision-Recall Curve for MIA on Purchases50 dataset.



(a)



(b)

Figure B.6: Precision-Recall Curve for MIA on Purchases100 dataset.

Bibliography

[1] Martín Abadi et al. "Deep learning with differential privacy." In: *Proceedings of the ACM Conference on Computer and Communications Security*. Vol. 24-28-Octo. 2016, pp. 308–318. ISBN: 9781450341394. DOI: 10.1145/2976749.2978318. URL: http://dx.doi.org/10.1145/2976749.2978318.

[2] Reza Shokri and Vitaly Shmatikov. "Privacy-Preserving Deep Learning." In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1310–1321. DOI: 10.1145/2810103.2813687.

[3] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions." In: (2019). URL: http://arxiv.org/abs/1908.07873.

[4] H Brendan McMahan Eider Moore Daniel Ramage Seth Hampson Blaise AgüeraAg and Agüera Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. Tech. rep. 2017.

[5] Daniel Bernau et al. *Assessing Differentially Private Deep Learning with Membership Inference*. Tech. rep. 2020.

[6] Rules O F Procedure. "Article 29 Data Protection Working Party." In: *October* Lx (2010). visited on 2020-08-02, pp. 1–8. URL: http://ec.europa.eu/justice/data-protection/index_en.htm.

[7] C Dwork et al. "The Algorithmic Foundations of Differential Privacy." In: *Foundations and Trends R in Theoretical Computer Science* 9 (2014), pp. 211–407. DOI: 10.1561/0400000042.

[8] *SAP Security Research*. visited on 2020-08-02. URL: https://www.sap.com/documents/2017/12/cc047065-e67c-0010-82c7-eda71af511fa.html.

[9] Milad Nasr, Reza Shokri, and Amir Houmansadr. *Comprehensive Privacy Analysis of Deep Learning Stand-alone and Federated Learning under Passive and Active White-box Inference Attacks*. Tech. rep. 2018.

[10] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. *Deep learning*. May 2015. DOI: 10.1038/nature14539.

[11] Balázs Csanád Csáji. *Approximation with Artificial Neural Networks*. Tech. rep. 2001.

[12] George Popov, Nikos Mastorakis, and Valeri Mladenov. *Calculation of the acceleration of parallel programs as a function of the number of threads*. Jan. 2010.

[13] Brendan McMahan and Research Scientists Daniel Ramage. *Google AI Blog: Federated Learning: Collaborative Machine Learning without Centralized Training Data*. visited on 2020-08-02. 2018. URL: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[14] Naman Agarwal et al. *cpSGD: Communication-efficient and differentially-private distributed SGD*. Tech. rep. 2018. arXiv: 1805.10559 [stat.ML].

[15] Nicolas Papernot et al. *Scalable Private Learning with PATE*. 2018. arXiv: 1802.08908 [stat.ML].

[16] Samuel Yeom et al. *Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting*. 2017. arXiv: 1709.01604 [cs.CR].

[17] Klas Leino and Matt Fredrikson. *Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference*. 2019. arXiv: 1906.11798 [cs.LG].

[18] Stacey Truex et al. *Demystifying Membership Inference Attacks in Machine Learning as a Service*. Tech. rep.

[19] Jinyuan Jia and Neil Zhenqiang Gong. *AttriGuard: A Practical Defense Against Attribute Inference Attacks via Adversarial Machine Learning*. ISBN: 978-1-939133-04-5. URL: www.usenix.org/conference/usenixsecurity18/presentation/jia-jinyuan.

[20] Ninghui Li et al. "Differential Privacy: From Theory to Practice." In: *Synthesis Lectures on Information Security, Privacy, and Trust* 8.4 (Oct. 2016), pp. 1–138. ISSN: 1945-9742.

[21] Frank Mcsherry. *Privacy Integrated Queries An Extensible Platform for Privacy-Preserving Data Analysis*. 2009. ISBN: 9781605585512.

[22] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. *The Composition Theorem for Differential Privacy*. 2013. arXiv: 1311.0776 [cs.DS].

[23] Ilya Mironov and Google Brain. *Rényi Differential Privacy*. Tech. rep. 2017.

[24] Shiva Prasad Kasiviswanathan et al. *What Can We Learn Privately? *.* Tech. rep. 2013.

[25] Jakub Konečny et al. *Federated Learning: Strategies for Improving Communication Efficiency*. Tech. rep.

[26] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning." In: (). DOI: 10.1145/3133956.3134012. URL: https://doi.org/10.1145/3133956.3134012.

[27] Hongyi Wang et al. *Federated Learning with Matched Averaging*. Tech. rep. 2020. URL: https://github.com/IBM/FedMA.

[28] Tian Li et al. *Federated Optimization in Heterogeneous Networks*. Tech. rep. 2020.

[29] Martín Abadi et al. "Deep Learning with Differential Privacy." In: (). DOI: 10.1145/2976749.2978318. URL: http://dx.doi.org/10.1145/2976749.2978318.

[30] Nan Wu et al. *The Value of Collaboration in Convex Machine Learning with Differential Privacy*. Tech. rep. 2019.

[31] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Basic Applications. Cambridge University Press, 2001. ISBN: 9780521830843. URL: https://books.google.de/books?id=tfzM9d%5C_jnxwC.

[32] Keith Bonawitz et al. *Towards Federated Learning at Scale: System Design*. 2019. arXiv: 1902.01046 [cs.LG].

[33] *Share Memory By Communicating - The Go Blog*. visited on 2020-08-02. URL: https://blog.golang.org/codelab-share.

[34] Michael Osborne. "Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature." In: 2010.

[35] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks." In: *Information Processing and Management* 45.4 (July 2009), pp. 427–437. ISSN: 03064573. DOI: 10.1016/j.ipm.2009.03.002.

[36] Jesse Davis and Mark Goadrich. "The Relationship between Precision-Recall and ROC Curves." In: ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 233–240. ISBN: 1595933832. DOI: 10.1145/1143844.1143874. URL: https://doi.org/10.1145/1143844.1143874.

[37] *tensorflow/privacy: Library for training machine learning models with privacy for training data*. visited on 2020-08-02. URL: https://github.com/tensorflow/privacy.

[38] Kaggle.com. *Acquire Valued Shoppers Challenge | Kaggle*. visited on 2020-08-02. 2014. URL: https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data.

[39] *Hospital Discharge Data Use Agreement*. visited on 2020-08-02. URL: https://www.dshs.texas.gov/THCIC/Hospitals/Download.shtm.

[40] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition." In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Mark W. Jones Xianghua Xie and Gary K. L. Tam. BMVA Press, Sept. 2015, pp. 41.1–41.12. ISBN: 1-901725-53-7. DOI: 10.5244/C.29.41. URL: https://dx.doi.org/10.5244/C.29.41.

[41] Max Ferguson et al. "Automatic localization of casting defects with convolutional neural networks Data Analytics for Smart Manufacturing Systems View project Neural Network Modeling for Prediction under Uncertainty in Energy System Applications View project Automatic Localization of Casting Defects with Convolutional Neural Networks." In: (2017). DOI: 10.1109/BigData.2017.8258115. URL: https://www.researchgate.net/publication/322512435.

[42] Liyue Fan. "Image pixelization with differential privacy." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10980 LNCS. Springer Verlag, July 2018, pp. 148–162. ISBN: 9783319957289. DOI: 10.1007/978-3-319-95729-6{_}10. URL: https://link.springer.com/chapter/10.1007/978-3-319-95729-6_10.

[43] Benjamin Zi et al. *On Inferring Training Data Attributes in Machine Learning Models*. Tech. rep. URL: https://www.tensorflow.org/guide/estimators.