Comprehensive Privacy Analysis of Deep Learning

Passive and Active White-box Inference Attacks against Centralized and Federated Learning

Milad Nasr
University of Massachusetts Amherst
milad@cs.umass.edu

Reza Shokri
National University of Singapore
reza@comp.nus.edu.sg

Amir Houmansadr
University of Massachusetts Amherst
amir@cs.umass.edu

***Abstract*—Deep neural networks are susceptible to various inference attacks as they *remember* information about their training data. We design white-box inference attacks to perform a comprehensive privacy analysis of deep learning models. We measure the privacy leakage through parameters of fully trained models as well as the parameter updates of models during training. We design inference algorithms for both centralized and federated learning, with respect to passive and active inference attackers, and assuming different adversary prior knowledge.**

We evaluate our novel *white-box membership inference attacks* against deep learning algorithms to trace their training data records. We show that a straightforward extension of the known black-box attacks to the white-box setting (through analyzing the outputs of activation functions) is ineffective. We therefore design new algorithms tailored to the white-box setting by exploiting the privacy vulnerabilities of the stochastic gradient descent algorithm, which is *the* algorithm used to train deep neural networks. We investigate the reasons why deep learning models may leak information about their training data. We then show that even well-generalized models are significantly susceptible to white-box membership inference attacks, by analyzing state-of-the-art pre-trained and publicly available models for the CIFAR dataset. We also show how adversarial participants, in the federated learning setting, can successfully run active membership inference attacks against other participants, even when the global model achieves high prediction accuracies.

I. Introduction

Deep neural networks have shown unprecedented generalization for various learning tasks, from image and speech recognition to generating realistic-looking data. This success has led to many applications and services that use deep learning algorithms on large-dimension (and potentially sensitive) user data, including user speeches, images, medical records, financial data, social relationships, and location data points. In this paper, we are interested in answering the following critical question: What is the privacy risk of deep learning algorithms to individuals whose data is used for training deep neural networks? In other words, how much is the information leakage of deep learning algorithms about their individual training data samples?

We define privacy-sensitive leakage of a model, about its training data, as the information that an adversary can learn from the model about them, which he is not able to infer from other models that are trained on other data from the same distribution. This distinguishes between the information that we can learn from the model about the data population, and the information that the model leaks about the particular data samples which are in its training set. The former indicates utility gain, and the later reflects privacy loss. We design inference attacks to quantify such privacy leakage.

Inference attacks on machine learning algorithms fall into two fundamental and related categories: *tracing* (a.k.a. membership inference) attacks, and *reconstruction* attacks [1]. In a reconstruction attack, the attacker's objective is to infer attributes of the records in the training set [2], [3]. In a membership inference attack, however, the attacker's objective is to infer if a particular individual data record was included in the training dataset [4], [5], [6]. This is a decisional problem, and its accuracy directly demonstrates the leakage of the model about its training data. We thus choose this attack as the basis for our privacy analysis of deep learning models.

Recent works have studied membership inference attacks against machine learning models in the *black-box* setting, where the attacker can only observe the model predictions [6], [7]. The results of these works show that the distribution of the training data as well as the generalizability of the model significantly contribute to the membership leakage. Particularly, they show that overfitted models are more susceptible to membership inference attacks than generalized models. Such black-box attacks, however, might not be effective against deep neural networks that generalize well (having a large set of parameters). Additionally, in a variety of real-world settings, the parameters of deep learning algorithms are visible to the adversaries, e.g., in a federated learning setting where multiple data holders collaborate to train a global model by sharing their parameter updates with each other through an aggregator.

Our contributions. In this paper, we present a comprehensive framework for the privacy analysis of deep neural networks, using white-box membership inference attacks. We go beyond membership inference attacks against fully-trained models. We take all major scenarios where deep learning is used for training and fine-tuning or updating models, with one or multiple collaborative data holders, when attacker only passively observes the model updates or actively influences the target model in order to extract more information, and for attackers with different types of prior knowledge. Despite differences in knowledge, observation, and actions of the adversary, their objective is the same: *membership inference*.

A simple extension of existing black-box membership inference attacks to the white-box setting would be using the same attack on all of the activation functions of the model. Our empirical evaluations show that this will not result in inference accuracy better than that of a black-box attacker. This is because the activation functions in the model tend to generalize much faster compared to the output layer. The early layers of a trained model extract very simple features that are not specific to the training data. The activation functions in the last layers extract complex and abstract features, thus should contain more information about the model's training set. However, this information is more or less the same as what the output leaks about the training data.

We design white-box inference attacks that **exploit the privacy vulnerabilities of the stochastic gradient descent (SGD) algorithm**. Each data point in the training set influences many of the model parameters, through the SGD algorithm, to minimize its contribution to the model's training loss. The local gradient of the loss on a target data record, with respect to a given parameter, indicates how much and in which direction the parameter needs to be changed to fit the model to the data record. To minimize the expected loss of the model, the SGD algorithm repeatedly updates model parameters in a direction that the gradient of the loss over the whole training dataset leans to *zero*. Therefore, **each training data sample will leave a distinguishable footprint on the gradients of the loss function over the model's parameters**.

We use the gradient vector of the model, over all parameters, on the target data point, as the main feature for the attack. We design deep learning attack models with an architecture that processes extracted (gradient) features from different layers of the target model separately, and combines their information to compute the membership probability of a target data point. We train the attack model for attackers with different types of background knowledge. Assuming a subset of the training set is known to the attacker, we can train the attack model in a supervised manner. However, for the adversary that lacks this knowledge, we train the attack model in an **unsupervised** manner. We train auto-encoders to compute a membership information embedding for any data. We then use a clustering algorithm, on the target dataset, to separate members from non-members based on their membership embedding.

To show the effectiveness of our white-box inference attack, **we evaluate the privacy of pre-trained and publicly available state-of-the-art models** on the CIFAR100 dataset. We had no influence on training these models. Our results show that the DenseNet model—which is the best model on CIFAR100 with 82% test accuracy—is not much vulnerable to black-box attacks (with a 54.5% inference attack accuracy, where 50% is the baseline for random guess). However, our white-box membership inference attack obtains a considerably higher accuracy of 74.3%. This shows that **even well-generalized deep models might leak significant amount of information about their training data, and could be vulnerable to white-box membership inference attacks**.

In federated learning, we show that a curious parameter server or even a participant can perform alarmingly accurate membership inference attacks against other participants. For the DenseNet model on CIFAR100, a local participant can achieve a membership inference accuracy of 72.2%, even though it only observes aggregate updates through the parameter server. Also, the curious central parameter server can achieve a 79.2% inference accuracy, as it receives the individual parameter updates from all participants. In federated learning, the *repeated* parameter updates of the models over different epochs on the *same* underlying training set is a key factor in boosting the inference attack accuracy.

As the contributions (i.e., parameter updates) of an adversarial participant can influence the parameters other parties, **in the federated learning setting, the adversary can actively push SGD to leak even more information about the participants' data**. We design an active attack that performs gradient *ascent* on a set of target data points before uploading and updating the global parameters. This magnifies the presence of data points in others' training sets, in the way SGD reacts by abruptly reducing the gradient on the target data points if they are members. On the Densenet model, this leads to a 76.7% inference accuracy for an adversarial participant, and a significant 82.1% accuracy for an active inference attack by the central server. By isolating a participant during parameter updates, the central attacker can boost his accuracy to 87.3%.

II. Inference Attacks

We use membership inference attacks to measure the information leakage through deep learning models about their training data. There are many different scenarios in which data is used for training models, and there are many different ways the attacker can observe the deep learning process. In Table I, we cover the major criteria to categorize the attacks. This includes attack observations, assumptions about the adversary knowledge, the target training algorithm, and the mode of the attack based on the adversary's actions. In this section, we discuss different attack scenarios as well as the techniques we use to exploit deep learning algorithms. We also describe the architecture of our attack model, and how the adversary computes the membership probability.

A. Attack Observations: Black-box vs. White-box Inference

The adversary's observations of the deep learning algorithm are what constitute the inputs for the inference attack.

Black-box. In this setting, the adversary's observation is limited to the output of the model on arbitrary inputs. For any data point $\mathbf{x}$, the attacker can only obtain $f(\mathbf{x};\mathbf{W})$. The parameters of the model $\mathbf{W}$ and the intermediate steps of the computation are *not* accessible to the attacker. This is the setting of machine learning as a service platforms. Membership inference attacks against black-box models are already designed, which exploit the statistical differences between a model's predictions on its training set versus unseen data [6].

White-box. In this setting, the attacker obtains the model $f(\mathbf{x};\mathbf{W})$ including its parameters which are needed for prediction. Thus, for any input $\mathbf{x}$, in addition to its output, the attacker can compute all the intermediate computations of the model. That is, the adversary can compute any function

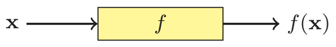
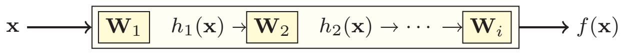
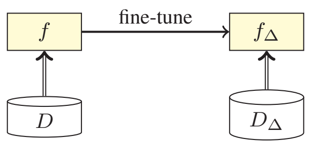
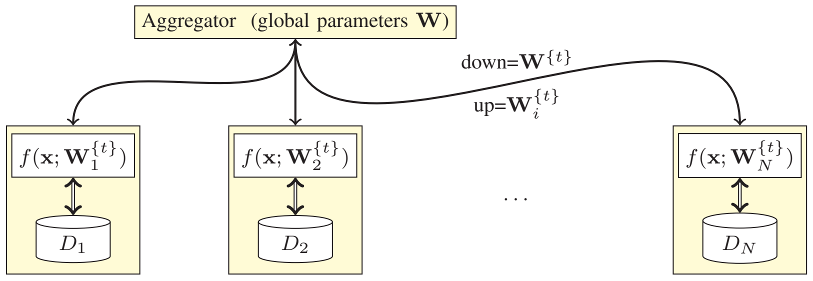
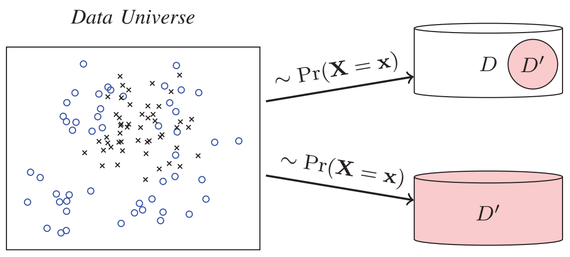
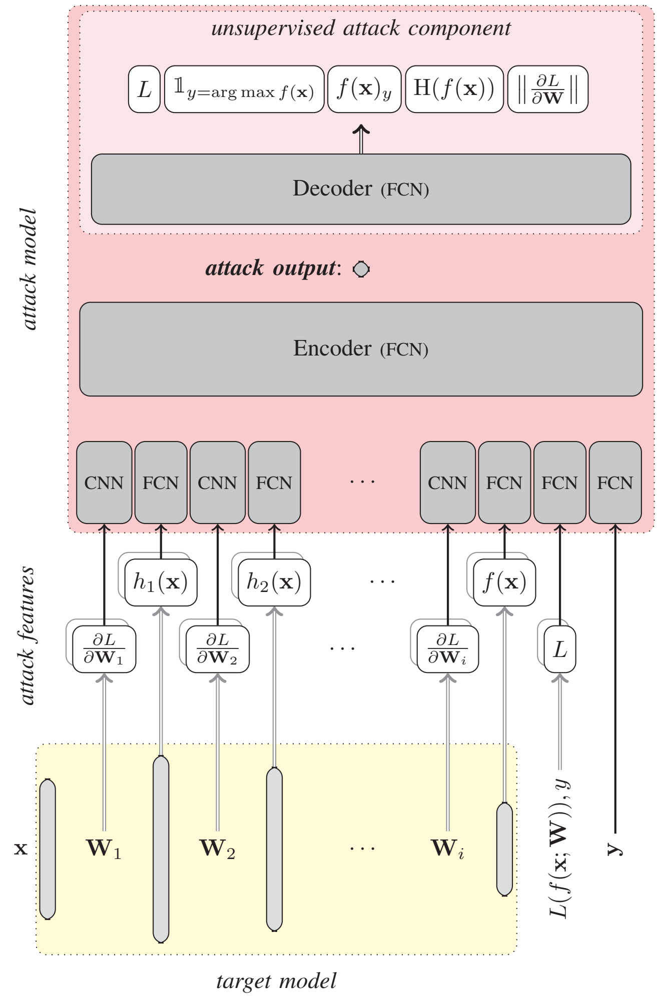
Criteria	**Attacks**	**Description**
Observation	Black-box	The attacker can obtain the prediction vector $f(\mathbf{x})$ on arbitrary input $\mathbf{x}$, but cannot access the model parameters, nor the intermediate computations of $f(\mathbf{x})$. 
	White-box	The attacker has access to the full model $f(\mathbf{x}; \mathbf{W})$, notably its architecture and parameters $\mathbf{W}$, and any hyper-parameter that is needed to use the model for predictions. Thus, he can also observe the intermediate computations at hidden layers $h_i(\mathbf{x})$.  $\mathbf{x}$ $\mathbf{W}_1$ $h_1(\mathbf{x})$ $\mathbf{W}_2$ $h_2(\mathbf{x}) \rightarrow \cdots$ $\mathbf{W}_i$ $f(\mathbf{x})$
Target	Stand-alone	The attacker observes the final target model f, after the training is done (e.g., in a centralized manner) using dataset D. He might also observe the updated model f_Δ after it has been updated (fine-tuned) using a new dataset D_Δ. 
	Federated	The attacker could be the central aggregator, who observes individual updates over time and can control the view of the participants on the global parameters. He could also be any of the participants who can observe the global parameter updates, and can control his parameter uploads.  Aggregator (global parameters $\mathbf{W}$) down=$\mathbf{W}^{\{t\}}$ up=$\mathbf{W}_i^{\{t\}}$ $f(\mathbf{x}; \mathbf{W}_1^{\{t\}})$ D_1 $f(\mathbf{x}; \mathbf{W}_2^{\{t\}})$ D_2 $\dots$ $f(\mathbf{x}; \mathbf{W}_N^{\{t\}})$ D_N
Mode	Passive	The attacker can only observe the genuine computations by the training algorithm and the model.
	Active	The attacker could be one of the participants in the federated learning, who adversarially modifies his parameter uploads $\mathbf{W}_i^{\{t\}}$, or could be the central aggregator who adversarially modifies the aggregate parameters $\mathbf{W}^{\{t\}}$ which he sends to the target participant(s).
Knowledge	Supervised	The attacker has a data set D', which contains a subset of the target set D, as well as some data points from the same underlying distribution as D that are not in D. The attacker trains an inference model h in a supervised manner, by minimizing the empirical loss function $\sum_{d \in D'} (1 - \mathbb{1}_{d \in D}) h(d) + \mathbb{1}_{d \in D}(1 - h(d))$, where the inference model h computes the membership probability of any data point d in the training set of a given target model f, i.e., $h(d) = \Pr(d \in D; f)$.  *Data Universe* $\sim \Pr(\mathbf{X} = \mathbf{x})$ D D' $\sim \Pr(\mathbf{X} = \mathbf{x})$ D'
	Unsupervised	The attacker has data points that are sampled from the same underlying distribution as D. However, he does not have information about whether a data sample has been in the target set D.

TABLE I: Various categories of inference attacks against machine learning models, based on their prior knowledge, observation, mode of attack, and the training architecture of the target models.



unsupervised attack component
L
$\mathbb{1}_{y=\arg\max f(\mathbf{x})}$
$f(\mathbf{x})_y$
$\mathrm{H}(f(\mathbf{x}))$
$\|\frac{\partial L}{\partial \mathbf{W}}\|$
Decoder (FCN)
attack output
Encoder (FCN)
attack model
CNN FCN CNN FCN ... CNN FCN FCN FCN
$h_1(\mathbf{x})$
$h_2(\mathbf{x})$
$f(\mathbf{x})$
$\frac{\partial L}{\partial \mathbf{W}_1}$
$\frac{\partial L}{\partial \mathbf{W}_2}$
$\frac{\partial L}{\partial \mathbf{W}_i}$
L
attack features
$\mathbf{x}$ $\mathbf{W}_1$ $\mathbf{W}_2$ $\mathbf{W}_i$
$L(f(\mathbf{x};\mathbf{W})), y$
$\mathbf{y}$
target model

Fig. 1: The architecture of our white-box inference attack. Given target data $(\mathbf{x}, y)$, the objective of the attack is to determine its membership in the training set D of target model f. The attacker runs the target model f on the target input $\mathbf{x}$, and computes all the hidden layers $h_i(\mathbf{x})$, the model's output $f(\mathbf{x})$, and the loss function $L(f(\mathbf{x}), y; \mathbf{W})$, in a forward pass. The attacker also computes the gradient of the loss with respect to the parameters of each layer $\frac{\partial L}{\partial \mathbf{W}_i}$, in a backward pass. These computations, in addition to the one-hot encoding of the true label $\mathbf{y}$, construct the input features of the inference attack. The attack model consists of convolutional neural network (CNN) components and fully connected network (FCN) components. For attacking federated learning and fine-tuning, the attacker observes each attack feature T times, and stacks them before they are passed to the corresponding attack component. For example, the loss features are composed as $L = \{L^{\{1\}}, L^{\{2\}}, \cdots, L^{\{T\}}\})$. The outputs of the CNN and FCN components are appended together, and this vector is passed to a fully connected encoder. The output of the encoder, which is a single value, is the attack output. This represents an embedding of the membership information in a single value. In the supervised attack setting, this embedding is trained to be $\Pr\{(\mathbf{x}, y) \in D\}$. In the unsupervised setting, a decoder is trained to reconstruct important features of the attack input (such as the model's output uncertainty $\mathrm{H}(f(\mathbf{x}))$, and the norm of its gradients $\|\frac{\partial L}{\partial \mathbf{W}}\|$) from the attack output. This is similar to deep auto-encoders. All unspecified attack layers are fully connected. The details of the architecture of the attack is presented in Table XIV in Appendix A.

over $\mathbf{W}$ and $\mathbf{x}$ given the model. The most straightforward functions are the outputs of the hidden layers, $h_i(\mathbf{x})$ on the input $\mathbf{x}$. As a simple extension, the attacker can extend black-box membership inference attacks (which are limited to the model's output) to the outputs of all activation functions of the model. However, this does not necessarily contain all the useful information for membership inference. Notably, the model output and activation functions could generalize if the model is well regularized. Thus, there might not be much difference, in distribution, between the activation functions of a model on its training versus unseen data. This can significantly limit the power of the inference attacks (as we also show in our evaluations).

What we suggest is to exploit the algorithm used to train deep learning models: the stochastic gradient descent (SGD) algorithm. Let $L(f(\mathbf{x}; \mathbf{W}), y)$ be the loss function for the classification model f. During the training, the SGD algorithm minimizes the empirical expectation of the loss function over the training set D:

$$\min_{\mathbf{W}} \mathbb{E}_{(\mathbf{x},y)\sim D}\big[L(f(\mathbf{x}; \mathbf{W}), y)\big] \tag{1}$$

The SGD algorithm solves this minimization by repeatedly updating parameters, $\mathbf{W}$, towards reducing the loss on small randomly selected subsets of D. Thus, for any data record in the training dataset, the gradient of the loss $\frac{\partial L}{\partial \mathbf{W}}$ over the data record is pushed towards zero, after each round of training. This is exactly what we can exploit to extract information about a model's training data.

For a target data record $(\mathbf{x}, y)$, the adversary can compute the loss of the model $L(f(\mathbf{x}; \mathbf{W}), y)$, and can compute the gradients of the loss with respect to all parameters $\frac{\partial L}{\partial \mathbf{W}}$ using a simple back-propagation algorithm. Given the large number of parameters used in deep neural networks (millions of parameters), the vector with such a significantly large dimension cannot properly generalize over the training data (which in many cases is an order of magnitude smaller in size). Therefore, the distribution of the model's gradients on members of its training data, versus non-members, is likely to be distinguishable. This can help the adversary to run an accurate membership inference attack, even though the classification model (with respect to its predictions) is well-generalized.

Inference model. We illustrate the membership inference attack in Figure 1. The significance of gradient (as well as activation) computations for a membership inference attack varies over the layers of a deep neural network. The first layers tend to contain less information about the specific data points in the training set, compared to non-member data record from the same underlying distribution. We can provide the gradients and activations of each layer as separate inputs to the attacker, as the attacker might need to design a specific attack for each layer. This enables the inference attack to split the inference task across different layers of the model, and then combine them to determine the membership. This engineering of the attack model architecture empowers the inference attack, as it reduces the capacity of the attack model and helps finding the optimal attack algorithm with less background data.

The distinct inputs to the attack model are the set of gradients $\frac{\partial L}{\partial \mathbf{W}_1}, \frac{\partial L}{\partial \mathbf{W}_2}, \cdots$, the set of activation vectors for different layers $h_1(\mathbf{x}), h_2(\mathbf{x}), \cdots$, the model output $f(\mathbf{x})$, the one-hot encoding of the label $\mathbf{y}$, and the loss of the model on the target data $L(f(\mathbf{x}; \mathbf{W}), y)$. Each of these are separately fed into the attack model, and are analyzed separately using independent components.

Inference attack components. The attack model is composed of feature extraction components and an encoder component. To extract features from the output of each layer, plus the one-hot encoding of the true label and the loss, we use fully connected network (FCN) submodules with one hidden layer. We use convolutional neural network (CNN) submodules for the gradients. When the gradients are computed on fully connected layers (in the target model), we set the size of the convolutional kernel to the input size of the fully connected layer, to capture the correlation of the gradients in each activation function. We reshape the output of each submodule component into a flat vector, and then concatenate the output of all components. We combine the outputs of all attack feature extraction components using a fully connected encoder component with multiple hidden layers. The output of the encoder is a single score, which is the output of the attack. This score (in the supervised attack raining) predicts the membership probability of the input data.

B. Inference Target: Stand-alone vs. Federated Learning

There are two major types of training algorithms for deep learning, depending on whether the training data is available all in one place (i.e., stand-alone centralized training), or it is distributed among multiple parties who do not trust each other (i.e., federated learning) [8]. In both cases, the attacker could be the entity who obtains the final trained model. In addition to such attack setting, the attacker might observe an updated version of the model after fine-tuning, for instance, which is very common in deep learning. Besides, in the case of federated learning, the attacker can be an entity who *participates* in the training. The settings of fine-tunning and federated learning are depicted in Table I.

Stand-alone fine-tunning. A model f is trained on dataset D. At a later stage it is updated to f_Δ after being fine-tuned using a new dataset D_Δ. If the attacker observes the final outcome, we want to measure the information leakage of the final model f_Δ about the whole training set $D \cup D_\Delta$. However, given that two versions of the model exist (before and after fine-tuning), we are also interested in measuring the extra information that could be learned about the training data, from the two model snapshots. The attacker might also be interested only in recovering information about the new set D_Δ. This is very relevant in numerous cases where the original model is trained using some unlabeled (and perhaps public) data, and then it is fine-tunned using sensitive private labeled data.

The model for inference attacks against fine-tunned models is a special case of our membership inference model for attacking federated learning. In both cases, the attacker observes multiple versions of the target model.

Federated learning. In this setting, N participants, who have different training sets D_i, agree on a single deep learning task and model architecture to train a global model. A central server keeps the latest version of the parameters W for the global model. Each participant has a local model, hence a local set of parameters W_i. In each epoch of training, each participant downloads the global parameters, updates them locally using SGD algorithm on their local training data, and uploads them back to the server. The parameter server computes the average value for each parameter using the uploaded parameters by all participants. This collaborative training continues until the global model converges.

There are two possibilities for the position of the attacker in federated learning: The adversary can be the centralized parameter server, or one of the participants. A curious parameter server can receive updates from each individual participant over time $W_i^{\{t\}}$, and use them to infer information about the training set of each participant. A malicious parameter server can also control the view of each participant on the global model, and can act actively to extract more information about the training set of a participant (as we discuss under active attacks). Alternatively, the adversary can be one of the participants. An adversarial participant can only observe the global parameters over time $W^{\{t\}}$, and craft his own adversarial parameter updates $W_i^{\{t\}}$ to gain more information about the union of the training data of all other participants.

In either of these cases, the adversary observes multiple versions of the target model over time. The adversary can try to run an independent membership inference attack on each of these models, and then combine their results. This, however, might not capture the dependencies between parameter values over time, which can leak information about the training data. Instead, in our design we make use of a single inference model, where each attack component (e.g., components for gradients of layer i) processes all of its corresponding inputs over the observed models at once. This is illustrated in Figure 1. For example, for the attack component that analyzes the loss value L, the input dimension can be $1 \times T$, if the adversary observes T versions of the target model over time. The output of the attack component is also T times larger than the case of attacking a stand-alone model. These correlated outputs, of all attack components, are processed all at once by the inference model.

C. Attack Mode: Passive vs. Active Inference Attack

The inference attacks are mostly passive, where the adversary makes observations without modifying the learning process. This is the case notably for attacking models after the training is over, e.g., the stand-alone setting.

Active attacks. The adversary, who is participating in the training process, can actively influence the target model in order to extract more information about its training set. This could be the case notably for running inference attacks against federated learning. In this setting, the central parameter server or a curious participant can craft adversarial parameter updates for a follow-up inference attack. The inference model architecture will be the same for passive and active attacks.

The active attacker can exploit the SGD algorithm to run the active attack. The insight we use to design our attack is that the SGD algorithm forcefully decreases the gradient of the loss on the training data, with the hope that this generalizes to the test data as well. The amount of the changes depends on the contribution of a data point in the loss. So, if a training data point leads to a large loss, the SGD algorithm will influence some parameters to adapt themselves towards reducing the loss on this point. If the data point is not seen by the model during training, the changes in the gradient on this point is gradual throughout the training. This is what we exploit in our active membership inference attack.

Let $\mathbf{x}$ be a data record, which is targeted by the adversary to determine its membership. Let us assume the adversary is one of the participants. The attacker runs a gradient *ascent* on $\mathbf{x}$, and updates its local model parameters in the direction of increasing the loss on $\mathbf{x}$. This can simply be done by adding the gradient to the parameters,

$$\mathbf{W} \leftarrow \mathbf{W} + \gamma \frac{\partial L^{\mathbf{x}}}{\partial \mathbf{W}}, \tag{2}$$

where γ is the adversarial update rate. The adversary then uploads the adversarially computed parameters to the central server, who will aggregate them with the parameter updates from other participants. The adversary can run this attack on a batch of target data points all at the same time.

If the target record $\mathbf{x}$ is in the training set of a participant, its local SGD algorithm abruptly reduces the gradient of the loss on $\mathbf{x}$. This can be detected by the inference model, and be used to distinguish members from non-members. Repeated active attacks, which happens in federated learning, lead to high confidence inference attacks.

D. Prior Knowledge: Supervised vs. Unsupervised Inference

To construct his inference attack model, the adversary needs to find the meaningful mapping between the model's behavior on a data point and its membership in the training set. The most straightforward way of learning such relationship is through some known members of the training data, and some data points from the same distribution which are not in the training data set. This is illustrated in Table I. The adversary has a dataset D' that overlaps with the target dataset D. Given this dataset, he can train the attack model in a supervised way, and use it to attack the rest of the training dataset.

Let h be the inference attack model. In the supervised setting, we minimize the (mean square) loss of the attacker for predicting the membership of the data points in its training set D':

$$\sum_{d \in D' \cap D} (h(d) - 1)^2 + \sum_{d \in D' \setminus D} (h(d))^2 \tag{3}$$

If the adversary does not have known samples from the target training set, there are two possibilities for training the inference attack models: supervised training on shadow models [6], and unsupervised training on the target model. Shadow models are models with the same architecture as the target model. The training data records for the shadow models are generated from the same distribution as the target training data, but do not have a known overlap with the target training set. The attacker trains the attack model on the shadow models. As the behavior of the shadow models on their training data is more or less similar to the behavior of the target model on its training data, the attack models trained on the shadow models are empirically shown to be effective.

The attack output for (shadow) supervised training setting is the probability of membership.

$$h(d) = \Pr(d \in D; f) \tag{4}$$

Unsupervised training of inference models. We introduce an alternative approach to shadow training, which is unsupervised training of the attack model on the target model. The assumption for this attack is that the attacker has access to a dataset D' which partially overlaps with the target training set D, however, the adversary does not know which data points are in $D' \cap D$.

Our objective is to find a score for each data point that represents its embedding in a space, which helps us easily separating members from non-members (using clustering algorithms). The attack's output should compute such representations. We make use of an encoder-decoder architecture to achieve this. This is very similar to the auto-encoders for unsupervised deep learning. As shown in Figure 1, the output of the attack is fed into a decoder. The decoder is a fully connected network with one hidden layer.

The objective of the decoder is to reconstruct some key features of the attack input which are important for membership inference. These include the loss value L, whether the target model has predicted the correct label $\mathbb{1}_{y=\arg\max f(\mathbf{x})}$, the confidence of the model on the correct label $f(\mathbf{x})_y$, the prediction uncertainty (entropy) of the model $\mathrm{H}(f(\mathbf{x}))$, and the norm of the gradients $\left\| \frac{\partial L}{\partial \mathbf{W}} \right\|$. As previous work [6] as well as our empirical results show, these features are strong signals for distinguishing members from non-members. The encoder-decoder architecture maximizes the information that the attack output contains about these features. Thus, it generates a membership embedding for each data point. Note that after training the attack model, the decoder plays no role in the membership inference attack.

The attack in the unsupervised setting is a batch attack, where the adversary attacks a large set of data records (disjoint from his background knowledge). We will use the encoder to for each target data record, and we compute the embedding value (output of the encoder model). Next, we use a clustering algorithm (e.g., we use the spectral clustering method) to cluster each input of the target model in two clusters. Note that the outcome of the clustering algorithm is a threshold, as the attack output is a single number. We predict the cluster with the larger gradient norm as non-members.

III. Experimental Setup

We implemented our attacks using Pytorch.[1] We trained all of the models on a PC equipped with four Titan X GPU each with 12 GB of memory.

A. Datasets

We used three datasets in our experiments: a standard image recognition benchmark dataset, CIFAR100, and two datasets Purchase100 and Texas100 [6].

CIFAR100. This is a popular benchmark dataset used to evaluate image recognition algorithms [9]. It contains $60,000$ color (RGB) images, each 32×32 pixels. The images are clustered into 100 classes based on objects in the images.

Purchase100. The Purchase100 dataset contains the shopping records of several thousand online customers, extracted during Kaggle's "acquire valued shopper" challenge.[2] The challenge was designed to identify offers that would attract new shoppers. We used the processed and simplified version of this dataset (courtesy of the authors of [6]). Each record in the dataset is the shopping history of a single user. The dataset contains 600 different products, and each user has a binary record which indicates whether she has bought each of the products (a total of $197,324$ data records). The records are clustered into 100 classes based on the similarity of the purchases, and our objective is to identify the class of each user's purchases.

Texas100. This dataset includes hospital discharge data records released by the Texas Department of State Health Services [3]. The records contain generic information about the patients (gender, age, and race), external causes of injury (e.g., drug misuse), the diagnosis, and patient procedures. Similar to Purchase100, we obtained the processed dataset (Courtesy of the authors [6]), which contains $67,330$ records and $6,170$ binary features.

B. Target Models

We investigate our attack model on the previously mentioned three datasets, Texas100, Purchase100 and CIFAR100. For the CIFAR100 dataset we used Alexnet [10], ResNet [11], DenseNet [12] models. We used SGD optimizer [13] to train the CIFAR100 models with learning rates of $0.01, 0.001, 0.0001$ for epochs $0-50, 50-100, 100-300$ accordingly. We used $l2$ regularization with weight of 0.0005.

For the Texas100 and Purchase100 datasets, we used fully connected models. For Purchase100, we used a model with layer sizes of $600, 1024, 512, 256, 128, 100$ (where 100 is the output layer), and for Texas100, we used layers with size $1024, 512, 256, 128, 100$ (where 100 is the output layer). We used Adam [13] optimizer with the learning rate of 0.001 for learning of these models. We trained each model for 100 epochs across all of our experiments. We selected the model with the best testing accuracy across all the 100 epochs.

[1]https://pytorch.org/

[2]https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data

[3]https://www.dshs.texas.gov/thcic/hospitals/Inpatientpudf.shtm

C. Pre-trained Models

To demonstrate that our attacks are not limited to our training algorithm, we used publicly available pre-trained CIFAR100 models[4]. All of these models are tuned to get the best testing accuracy using different regularization techniques.

D. Federated Learning

We performed the training for all of the federated learning experiments. Specifically, we used the averaging aggregation method for the federated scenario [8]. Each training party sends the parameter updates after every epoch of training to the central model, and the central server averages the models' updates from the parties and sends the updated model to all parties. In our experiments, we use the same training dataset size for all parties, and each party's training data is selected uniformly at random from our available datasets.

E. Attack Models

Table XIV in Appendix A, presents the details of our attack model architecture. As can be seen, we used ReLU activation functions, and we initialized the weights using a normal distribution with mean 0 and standard deviation of 0.01. The bias values of all layers are initialized with 0. The batch size of all experiments is 64. To train the attack model we use the Adam optimizer with a learning rate of 0.0001. We train attack models for 100 epochs and pick the model with the highest testing accuracy, across all the 100 epochs.

Tables II and XI present the dataset sizes used for training the target and attack models. In the supervised setting for training the attack models, we assume the attacker has access to a fraction of the training set and some non-member samples. In this case, to balance the training, we select half of each batch to include member instances and the other half non-member instances from the attacker's background knowledge. Creating the batches in this fashion will prevent the attack model from a bias towards member or non-member instances.

F. Evaluation Metrics

Attack accuracy The attacker's output has two classes "*Member*" and "*Non-member*". Attack accuracy is the fraction of the correct membership predictions (predicting members as member and non-members as non-member) for unknown data points. The size of the set of member and non-member samples that we use for evaluating the attack are the same.

True/False positive For a more detailed evaluation of attack performance, we also measure the true positive and false positive rates of the attacker. Positive is associated with the attacker outputting "member".

Prediction uncertainty For a classification model, we compute its prediction uncertainty using the normalized entropy of its prediction vector for a given input.

$$\mathrm{H} = \frac{1}{\log(K)} \sum_{i=1}^{K} p_i \log(p_i) \tag{5}$$

[4]We make use of ResNet, DenseNet, and Alexnet pre-trained models, provided in https://github.com/bearpaw/pytorch-classification

TABLE II: Size of datasets used for training and testing the target classification model and the membership inference model

Datasets	**Target Model**		**Inference Attack Model**			
	Training	**Test**	**Training Members**	**Training Non-members**	**Test Members**	**Test Non-members**
CIFAR100	50,000	10,000	25,000	5,000	5,000	5,000
Texas100	10,000	70,000	5,000	10,000	10,000	10,000
Puchase100	20,000	50,000	10,000	10,000	10,000	10,000

where K is the number of all classes and p_i is the prediction probability for the ith class. We compute the probabilities using a softmax function as $p_i = \frac{e^{h(d)^{(i)}}}{\sum_{k=1}^{K} e^{h(d)^{(k)}}}$.

IV. Experiments

We start by presenting our results for the stand-alone scenario, followed by our results for the federated learning scenario.

A. Stand-Alone Setting: Attacking Fully-Trained Models

We investigate the case where the attacker has access to the fully-trained target model, in the white-box setting. Therefore, the attacker can leverage the outputs and the gradients of the hidden layers of the target model to perform the attack. We have used pre-trained CIFAR100 models, and have trained other target models and the attack models using the dataset sizes which are presented in Table II.

Impact of the outputs of different layers: To understand and demonstrate the impact of different layers' outputs, we perform the attack separately using the outputs of *individual layers*. We use a pre-trained Alexnet model as the target model, where the model is composed of five convolutional layers and a fully connected layer at the end. Table III shows the accuracy of the attack using the output of each of the last three layers. As the table shows, using the last layers results in the highest attack accuracy, i.e., **among the layer outputs, the last layer (model output) leaks the most membership information about the training data**.The reason behind this is twofold. By proceeding to the later layers, the capacity of the parameters ramps up, which leads the target model to store unnecessary information about the training dataset, and therefore leak more information. Moreover, the first layers extract simple features from the input, thus generalize much better compared to the last layers, which are responsible for complex task of finding the relationship between abstract features and the classes. We did not achieve significant accuracy gain by combining the outputs from multiple layers; this is because the leakage from the last layer (which is equivalent to a black-box inference attack) already contains the membership information that leaks from the output of the previous layers.

Impact of gradients: In Section II-A, we discussed why gradients should leak information about the training dataset. In Table VIII, we compare the accuracy of the membership attack when the attacker uses the gradients versus layer outputs, for different dataset and models. Overall, the results show that *gradients leak significantly more membership information about the training set, compared to the layer outputs*.

We compare the result of the attack on pre-trained CIFAR100-ResNet and CIFAR100-DenseNet models, where both are designed for the same image recognition task, both are trained on the same dataset, and both have similar generalization error. The results show that these two models have various degrees of membership leakage; this suggests that the **generalization error is not the right metric to quantify privacy leakage in the white-box setting. The large capacity of the model which enables it to learn complex tasks and generalize well, leads to also memorizing more information about the training data.** The total number of the parameters in pre-trained Densenet model is 25.62M , whereas this is only 1.7M parameters for ResNet.

We also investigated the impact of gradients of different layers on attack accuracy. The results are shown in Table IV show that **the gradient of the later layers leak more membership information**. This is similar to our findings for layer outputs: the last layer generalizes the least among all the layers in the model, and is the most dependent layer to the training set. By combining the gradients of all layers, we are able to only slightly increase the attack accuracy.

Finally, Table V shows the attack accuracy when we combine the output layer and gradients of different layers. We see that the gradients from the last layer leak the most membership information.

Impact of the training size: Table VI shows attack accuracy for various sizes of the attacker's training data. The models are tested on the same set of test instances, across all these scenarios. As expected, increasing the size of the attacker's training dataset improves the accuracy of the membership inference attack.

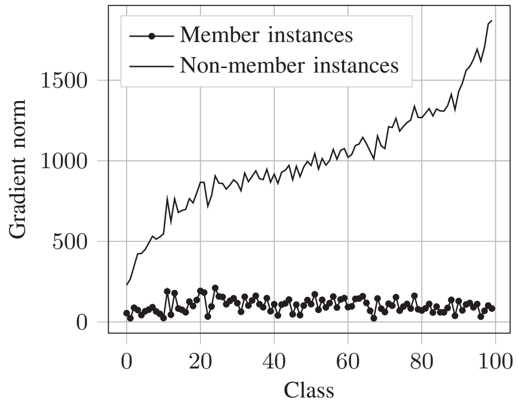
Impact of the gradient norm: In this experiment, we demonstrate that the norm of the model's gradients is highly correlated with the accuracy of membership inference, as it behaves differently for member and non-member instances. Figure 3 plots the last-layer gradient norms over consecutive training epochs for member and non-member instances (for the Purchase100 dataset). As can be seen, during training, the gradient norms of the member instances decrease over training epochs, which is not the case for non-member instances.

Figure 4 shows the distribution of last-layer gradient norms for members and non-members on three various pretrained architectures on CIFAR100. Comparing the figures with Table VIII, we see that *a model leaks more membership information when the distribution of the gradient norm is more distinct for member and non-member instances*. For instance, we can see that ResNet and DenseNet both have relatively similar generalization errors, but the gradient norm distribution of members and non-members is more distinguishable for DenseNet (Figure 4b) compared to ResNet (Figure 4c). We see that the attack accuracy in DenseNet is much higher than ResNet.

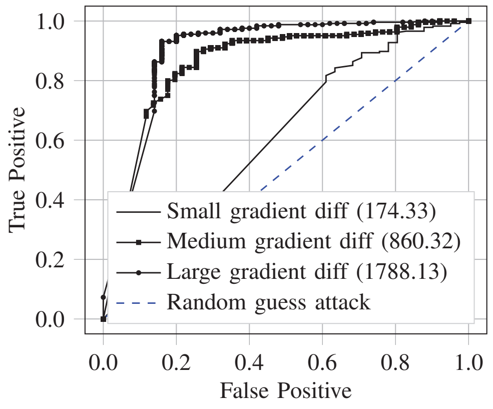
Also, we show that the accuracy of our inference attack is

TABLE III: Attack accuracy using the outputs of individual activation layers. Pre-trained Alexnet on CIFAR100, stand-alone setting.

Output Layer	Attack Accuracy
Last layer (prediction vector)	74.6% (black-box)
Second to last	74.1%
Third to last	72.9%
Last three layers, combined	74.6%



(a) Gradient norm for member and non-member data across all classes



(b) Attacker accuracy for class members with various differences in their member/non-member gradient norms

Fig. 2: Attack accuracy is different for different output classes (pre-trained CIFAR100-Alextnet model in the stand-alone scenario).

higher for classification output classes (of the target model) with a larger difference in member/non-member gradient norms. Figure 2a plots the average of last layer's gradient norms for different output classes for member and non-member instances; we see that the difference of gradient norms between members and non-members varies across different classes. Figure 2b shows the receiver operating characteristic (ROC) curve of the inference attack for three output classes with small, medium, and large differences of gradient norm between members and non-members (averaged over many samples). As can be seen, *the larger the difference of gradient norm between members and non-members, the higher the accuracy of the membership inference attack*.

TABLE IV: Attack accuracy when we apply the attack using parameter gradients of different layers. (CIFAR100 dataset with Alexnet architecture, stand-alone scenario)

Gradient w.r.t.	Attack Accuracy
Last layer parameters	75.1%
Second to last layer parameters	74.6%
Third to last layer parameters	73.5%
Forth to last layer parameters	72.6%
Parameters of last four layers, combined	75.15%

TABLE V: Attack accuracy using different combinations of layer gradients and outputs. (CIFAR100 dataset with Alexnet architecture, stand-alone scenario)

Gradients w.r.t.	Output Layers	Attack Test Accuracy
Last Layer	-	75.10%
Last layer	Last layer	75.11%
Last Layer	All Layer	75.12%
All Layer	All Layer	75.18%

TABLE VI: Attack accuracy for various sizes of the attacker's training dataset. The size of the target model's training dataset is 50,000. (The CIFAR100 dataset with Alexnet, stand-alone scenario)

Members Sizes	Non-members Sizes	Attack Accuracy
10,000	2,000	73.2%
15,000	2,000	73.7%
15,000	5,000	74.8%
25,000	5,000	75.1%

TABLE VII: Accuracy of our unsupervised attack compared to the Shadow models approach [6] for the white-box scenario.

Dataset	Arch	(Unsupervised) Attack Accuracy	(Shadow Models) Attack Accuracy
CIFAR100	Alexnet	75.0%	70.5%
CIFAR100	DenseNet	71.2%	64.2%
CIFAR100	ResNet	63.1%	60.9%
Texas100	Fully Connected	66.3%	65.3%
Purchase100	Fully Connected	71.0%	68.2%

Impact of prediction uncertainty: Previous work [6] claims that the prediction vector's uncertainty is an important factor in privacy leakage. We validate this claim by evaluating the attack for different classes in CIFAR100-Alexnet with different prediction uncertainties. Specifically, we selected three classes with small, medium, and high differences of prediction uncertainties, where the attack accuracies are shown in Figure 6 for these classes. Similar to the differences in gradient norms, *the classes with higher prediction uncertainty values leak more membership information*.

B. Stand-Alone Setting: Unsupervised Attacks

We also implement our attacks in an unsupervised scenario, in which the attacker has data points sampled from the same

TABLE VIII: The attack accuracy for different datasets and different target architectures using layer outputs versus gradients. This is the result of analyzing the stand-alone scenario, where the CIFAR100 models are all obtained from pre-trained online repositories.

Target Model				Attack Accuracy		
Dataset	**Architecture**	**Train Accuracy**	**Test Accuracy**	**Black-box**	**White-box (Outputs)**	**White-box (Gradients)**
CIFAR100	Alexnet	99%	44%	74.2%	74.6%	75.1%
CIFAR100	ResNet	89%	73%	62.2%	62.2%	64.3%
CIFAR100	DenseNet	100%	82%	67.7%	67.7%	74.3%
Texas100	Fully Connected	81.6%	52%	63.0%	63.3%	68.3%
Purchase100	Fully Connected	100%	80%	67.6%	67.6%	73.4%

TABLE IX: Attack accuracy on fine-tuned models. D is the initial training set, $D_\triangle$ is the new dataset used for fine-tuning, and $\bar{D}$ is the set of non-members (which is disjoint with D and $D_\triangle$).

Dataset	**Architecture**	**Train Acc.**	**Test Acc.**	**Distinguish D from $D_\triangle$**	**Distinguish D from $\bar{D}$**	**Distinguish $D_\triangle$ from $\bar{D}$**
CIFAR100	Alexnet	100.0%	39.8%	62.1%	75.4%	71.3%
CIFAR100	DenseNet	100.0%	64.3%	63.3%	74.6%	71.5%
Texas100	Fully Connected	95.2%	48.6%	58.4%	68.4%	67.2%
Purchase100	Fully Connected	100.0%	77.5%	64.4%	73.8%	71.2%

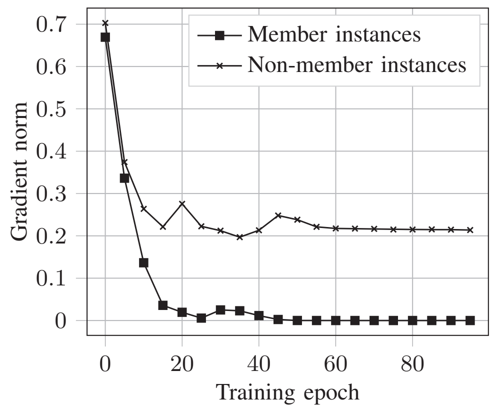


Fig. 3: Gradient norms of the last layer during learning epochs for member and non-member instances (for Purchase100).

underlying distribution, but he does not know their member and non-member labels. In this case, the attacker classifies the tested records into two clusters as described in Section II-D.

We implemented our attack and compared its performance to Shadow models of Shokri et al. [6] introduced earlier. We train our unsupervised models on various datasets based on the training and test dataset sizes in Table II. We train a single Shadow model on each of Texas100 and Purchase100 datasets using training sizes according to Table II. The training sets of the Shadow models do no overlap with the training sets of the target models. For the CIFAR100 dataset, however, our Shadow model uses a training dataset that overlaps with the target model's dataset, as we do not have enough instances (we train each model with 50,000 instances out of the total 60,000 available records).

After the training, we use the Spectral clustering algorithm [14] to divide the input samples into two clusters. As shown earlier (Figure 4), the member instances have smaller gradient norm values. Therefore, we assign the member label to the cluster with a smaller average gradient norm, and the non-member label to the other cluster.

Table VII compares the accuracy of our unsupervised attack with shadow training [6] on various datasets and architectures. We see that *our approach offers a noticeably higher accuracy*.

The intuition behind our attack working is that the encoded values of our unsupervised algorithm present different distributions for member and non-member samples. This can be seen in Figure 5 for various datasets and architectures.

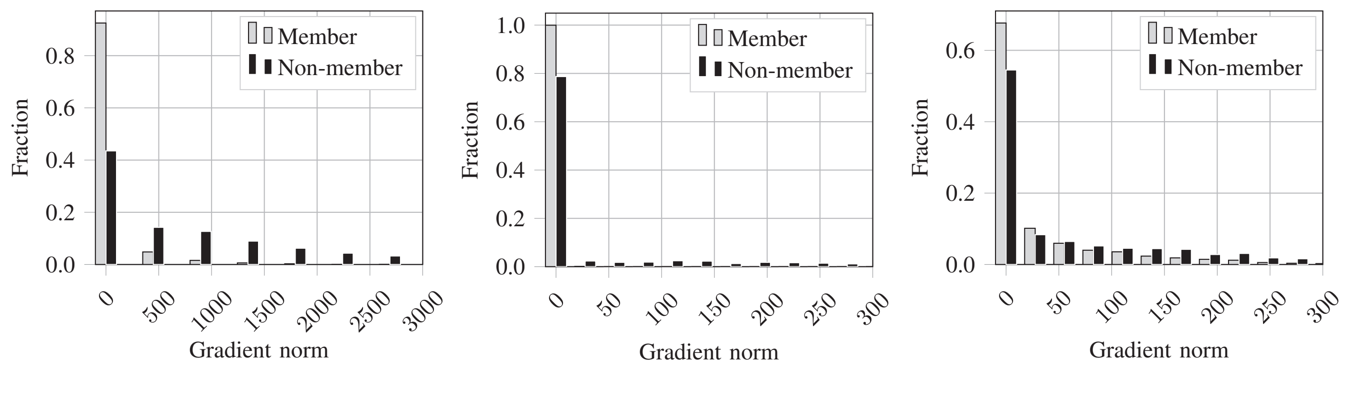
C. Stand-Alone Setting: Attacking Fine-Tuned Models

We investigate privacy leakage of fine-tuned target models. In this scenario, the victim trains a model with dataset D, then he uses a dataset $D_\triangle$ to fine-tune the trained model to improve its performance. Hence, the attacker has two snapshots of the trained model, one using only D, and one for the same model which is fine-tuned using $D_\triangle$. We assume the attacker has access to both of the trained models (before and after fine-tuning). We are interested in applying the membership inference attack in this scenario, where the goal of the adversary is to distinguish between the members of D, $D_\triangle$, and $\bar{D}$, which is a set of non-members.

We use the same training dataset as in the previous experiments (Table II); we used 60% of the train dataset as D and the rest for $D_\triangle$. Table IX shows the train, test, and attack accuracy for different scenarios. As can be seen, the attacker is able to distinguish between members (in D or $D_\triangle$) and non-members ($\bar{D}$) with accuracies similar to previous settings. Additionally, the attacker can also distinguish between the members of D and $D_\triangle$ with reasonably high accuracies.

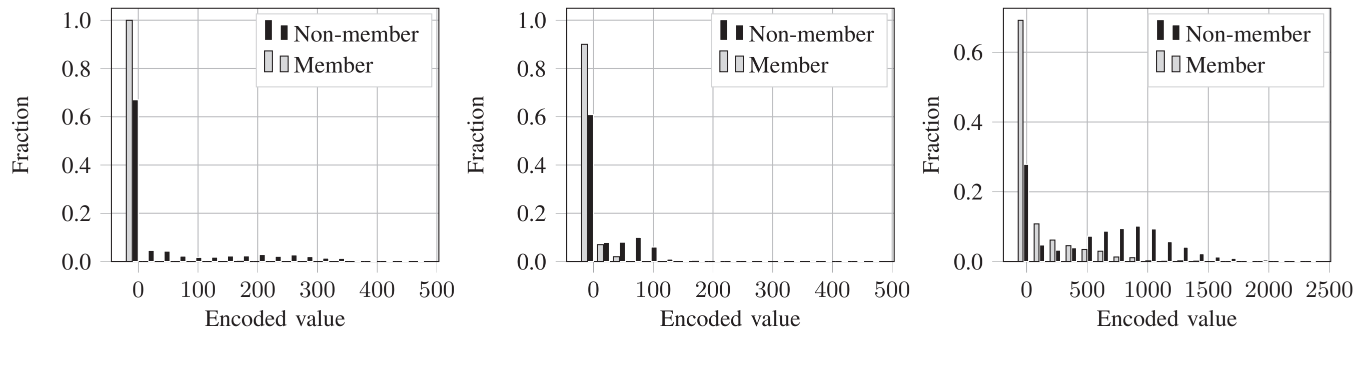
D. Federated Learning Settings: Passive Inference Attacks

Table XI shows the dataset sizes used in our federated attack experiments. For the CIFAR100 experiment with a



(a) CIFAR100-Alexnet (b) CIFAR100-Densenet (c) CIFAR100-Resnet

Fig. 4: The distribution of gradient norms for member and non-member instances of different pretrained models.



(a) DenseNet-CIFAR100 (b) ResNet-CIFAR100 (c) AlexNet-CIFAR100

Fig. 5: The distribution of the encoded values (i.e., the attack output) for the member and non-member instances of our unsupervised algorithm are distinguishable. This is the intuition behind the high accuracy of our unsupervised attack.

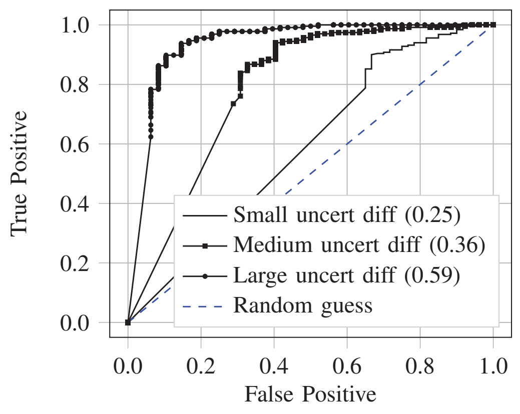


Fig. 6: Attack's ROC for three different classes of data with large, medium, and small prediction uncertainty values (pre-trained CIFAR100-Alextnet model in the stand-alone scenario).

local attacker, each participant uses 30,000 instances to train, which overlaps between various participants due to non-sufficient number of instances. For all the other experiments, the participants use non-overlapping datasets. In the following, we present the attack in various settings.

The Passive Global Attacker: In this scenario, the attacker (the parameter aggregator) has access to the target model's parameters over multiple training epochs (see Section II-B). Thus, he can passively collect all the parameter updates from all of the participants, at each epoch, and can perform the membership inference attack against each of the target participants, separately.

Due to our limited GPU resources, our attack observes each target participant during only five (non-consecutive) training epochs. Table XII shows the accuracy of our attack when it uses different sets of training epochs (for the CIFAR100 dataset with Alexnet). We see that *using later epochs, substantially increases the attack accuracy*. Intuitively, this is because the earlier training epochs contain information of the generic features of the dataset, which do not leak significant membership information, however, the later epochs contain more membership information as the model starts to learn the outliers in such epochs [15].

Table X presents the results of this attack on different datasets. For the Purchase100 and Texas100 datasets we use the $[40, 60, 80, 90, 100]$ training epochs, and for the CIFAR100

TABLE X: Attack accuracy in the federated learning setting. There are 4 participants. A global attacker is the central parameter aggregator, and the local attacker is one of the participants. The global attacker performs the inference against each individual participant, and we report the average attack accuracy. The local attacker performs the inference against all other participants. The passive attacker follows the protocol and only observes the updates. The active attacker changes its updates, or (in the case of a global attack) isolates one participant by not passing the updates of other participants to it, in order to increase the information leakage.

Target Model		**Global Attacker (the parameter aggregator)**				**Local Attacker (a participant)**	
		Passive	Active			Passive	Active
Dataset	Architecture		Gradient Ascent	Isolating	Isolating Gradient Ascent		Gradient Ascent
CIFAR100	Alexnet	85.1%	88.2%	89.0%	92.1%	73.1%	76.3%
CIFAR100	DenseNet	79.2%	82.1%	84.3%	87.3%	72.2%	76.7%
Texas100	Fully Connected	66.4%	69.5%	69.3%	71.7%	62.4%	66.4%
Purchase100	Fully Connected	72.4%	75.4%	75.3%	82.5%	65.8%	69.8%

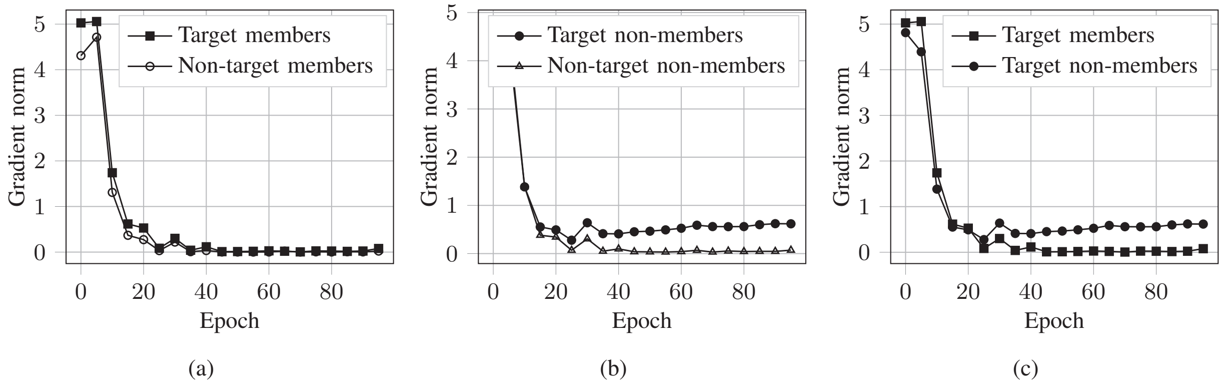


Fig. 7: The impact of the global active gradient ascent attack on the target model's training process. Figures show the gradient norms of various instances (Purchase100 dataset) during the training phase, while the target instances are under attack.

TABLE XI: Dataset sizes in the federated learning experiments

Datasets	**Parties' Datasets**		**Inference Attack Model**			
	Training	**Test**	**Training Members**	**Training Non-members**	**Test Members**	**Test Non-members**
CIFAR100	30,000	10,000	15,000	5,000	5,000	5,000
Texas100	8,000	70,000	4,000	4,000	4,000	4,000
Puchase100	10,000	50,000	5,000	5,000	5,000	5,000

TABLE XII: The accuracy of the passive global attacker in the federated setting when the attacker uses various training epochs. (CIFAR100-Alexnet)

Observed Epochs	**Attack Accuracy**
$5, 10, 15, 20, 25$	57.4%
$10, 20, 30, 40, 50$	76.5%
$50, 100, 150, 200, 250$	79.5%
$100, 150, 200, 250, 300$	85.1%

TABLE XIII: The accuracy of the passive local attacker for different numbers of participants. (CIFAR100-Alexnet)

Number of Participants	**Attack Accuracy**
2	89.0%
3	78.1%
4	76.7%
5	67.2%

dataset we use epochs $[100, 150, 200, 250, 300]$. When the attacker has access to several training epochs in the CIFAR100 target models, he achieves a high membership attack accuracy. In Texas100 and Purchase100 datasets, however, the accuracy of the attack decreases compare to the stand-alone setting. This is due to the fact that *averaging in the federated learning scenarios will reduce the impact of each individual party*.

The Passive Local Attacker: A local attacker cannot observe the model updates of the participants; he can only observe the *aggregate* model parameters. We use the same attack model architecture as that of the global attack. In our experiments, there are four participants (including the local attacker). The goal of the attacker is to learn if a target input has been a member of the training data of any other participants. Table X shows the accuracy of our attack on various datasets. As expected, a local attack has a lower accuracy compared to the global attack; this is because the local attacker observes the *aggregate* model parameters of all participants, which limits the extent of membership leakage. *The accuracy of the local attacker degrades for larger numbers of participants*. This is shown in Table XIII for the CIFAR100 on Alexnet model.

E. Federated Learning Settings: Active Inference Attacks

Table X shows the results of attacks on federated learning.

The Gradient Ascent Attacker: In this scenario, the attacker adversarially manipulates the learning process to improve the membership inference accuracy. The active attack is described in Section II-C. We evaluate the attack accuracy on predicting the membership of 100 randomly sampled member instances, from the target model, and 100 non-member instances. For all such target instances (whose membership is unknown to the attacker), the attacker updates their data features towards ascending the gradients of the global model (in case of the global attack) or the local model (in the case of a local attack).

Figure 7 compares the last-layer gradient norm of the target model for different data points. As Figure 7a shows, when the attacker ascends on the gradients of the target instances, the gradient norm of the target members will be very similar to that of non-target member instances in various training epochs. On the other hand, this is not true for the non-member instances as shown in Figure 7b.

Intuitively, this is because applying the gradient ascent algorithm on a member instance will trigger the target model to try to minimize its loss by descending in the direction of the model's gradient for those instances (and therefore nullify the effect of the attacker's ascent). For target non-member instances, however, the model will not explicitly change their gradient, as they do not influence the training loss function. The attacker repeats gradient ascend algorithm for each epoch of the training, therefore, the gradient of the model will keep increasing on such non-member instances. Figure 7c depicts the resulted distinction between the gradient norm of the member and non-member target instances. The active gradient ascend attacker forces the target model to behave drastically different between target member and target non-member instances which makes the membership inference attack easier. As a result, compared to the passive global attacker we see that the active attack can noticeably gain higher accuracy. In the local case, the accuracy is lower than the global attack due to the observation of aggregated parameters from multiple participants.

The Isolating Attacker: The parameter aggregation in the federated learning scenario negatively influences the accuracy of the membership inference attacks. An active global attacker can overcome this problem by isolating a target participant, and creating a local view of the network for it. In this scenario, the attacker does not send the aggregate parameters of all parties to the target party. Instead, the attacker isolates the target participant and segregates the target participant's learning process.

When the attacker isolates the target participant, then the target participant's model does not get aggregated with the parameters of other parties. As a result, it stores more information about its training dataset in the model. Thus, simply *isolating the training of a target model significantly increases the attack accuracy*. We can apply the isolating method to the gradient ascent attacker and further improve the attacker accuracy. See Table X for all the results.

V. Related Work

Investigating different inference attacks on deep neural networks is an active area of research.

A. Membership Inference Attacks

Multiple research papers have studied membership inference attacks in a black-box setting [6], [16], [7]. Homer et al. [4] performed one of the first membership inference attacks on genomic data. Shokri et al. [6] showed that an ML model's output has distinguishable properties about its training data, which could be exploited by the adversary's inference model. They introduced shadow models that mimic the behavior of the target model, which are used by the attacker to train the attack model. Salem et al. [17] extended the attacks of Shokri et al. [6] and showed empirically that it is possible to use a single shadow model (instead of several shadow models used in [6]) to perform the same attack. They further demonstrated that even if the attacker does not have access to the target model's training data, she can use statistical properties of outputs (e.g., entropy) to perform membership inference. Yeom et al. [7] demonstrated the relationship between overfitting and membership inference attacks. Hayes et al. [18] used generative adversarial networks to perform membership attacks on generative models.

Melis et al. [19] developed a new set of membership inference attacks for the collaborative learning. The attack assumes that the participants update the central server after each mini-batch, as opposed to updating after each training epoch [20], [21]. Also, the proposed membership inference attack is designed exclusively for models that use explicit word embeddings (which reveal the set of words used in the training sentences in a mini-batch) with very small training mini-batches.

In this paper, we evaluate standard learning mechanisms for deep learning and standard target models for various architectures. We showed that our attacks work even if we use pre-trained, state-of-the-art target models.

Differential privacy [22], [23] has been used as a strong defense mechanism against inference attacks in the context of machine learning [24], [25], [26], [27]. Several works [28], [29], [30] have shown that by using adversarial training, one can find a better trade-off between privacy and model accuracy. However, the focus of this line of work is on the membership inference attack in the black-box setting.

B. Other Inference Attacks

An attacker with additional information about the training data distribution can perform various types of inference attacks. Input inference [31], attribute inference [32], parameter inference [33], [34], and side-channel attacks [35] are several examples of such attacks. Ateniese et al. [36] show that an adversary with access to the parameters of machine learning models such as Support Vector Machines (SVM) or Hidden Markov Models (HMM) [37] can extract valuable information about the training data (e.g., the accent of the speakers in speech recognition models).

VI. Conclusions

We designed and evaluated novel white-box membership inference attacks against neural network models by exploiting the privacy vulnerabilities of the stochastic gradient descent algorithm. We demonstrated our attacks in the stand-alone and federated settings, with respect to passive and active inference attackers, and assuming different adversary prior knowledge. We showed that even well-generalized models are significantly susceptible to such white-box membership inference attacks. Our work did not investigate theoretical bounds on the privacy leakage of deep learning in the white-box setting, which would remain as a topic of future research.

Acknowledgements

This work was supported in part by the NSF grant CNS-1525642, as well as the Singapore Ministry of Education Academic Research Fund Tier 1, R-252-000-660-133. Reza Shokri would like to acknowledge the support of NVIDIA Corporation with the donation of a Titan Xp GPU which was used for this research.

References

[1] C. Dwork, A. Smith, T. Steinke, and J. Ullman, "Exposed! a survey of attacks on private data," 2017.

[2] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2003, pp. 202–210.

[3] R. Wang, Y. F. Li, X. Wang, H. Tang, and X. Zhou, "Learning your identity and disease from research papers: information leaks in genome wide association study," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 534–544.

[4] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. V. Pearson, D. A. Stephan, S. F. Nelson, and D. W. Craig, "Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays," *PLoS genetics*, vol. 4, no. 8, p. e1000167, 2008.

[5] C. Dwork, A. Smith, T. Steinke, J. Ullman, and S. Vadhan, "Robust traceability from trace amounts," in *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE, 2015, pp. 650–669.

[6] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*, 2017.

[7] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *IEEE Computer Security Foundations Symposium*, 2018.

[8] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[9] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, 2016, vol. 1.

[14] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[15] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[16] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *arXiv preprint arXiv:1802.04889*, 2018.

[17] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[18] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: evaluating privacy leakage of generative models using generative adversarial networks," *arXiv preprint arXiv:1705.07663*, 2017.

[19] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," *arXiv preprint arXiv:1805.04049*, 2018.

[20] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.

[21] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*. Springer, 2006, pp. 265–284.

[23] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[24] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011.

[25] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE, 2014, pp. 464–473.

[26] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[27] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with pate," *arXiv preprint arXiv:1802.08908*, 2018.

[28] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 634–646.

[29] J. Hamm, "Minimax filter: learning to preserve privacy from inference attacks," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 4704–4734, 2017.

[30] C. Huang, P. Kairouz, X. Chen, L. Sankar, and R. Rajagopal, "Generative adversarial privacy," *arXiv preprint arXiv:1807.05306*, 2018.

[31] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[32] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: Measuring unintended neural network memorization & extracting secrets," *arXiv preprint arXiv:1802.08232*, 2018.

[33] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *USENIX Security*, 2016.

[34] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," *arXiv preprint arXiv:1802.05351*, 2018.

[35] L. Wei, Y. Liu, B. Luo, Y. Li, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," *arXiv preprint arXiv:1803.05847*, 2018.

[36] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[37] C. Robert, *Machine learning, a probabilistic perspective*. Taylor & Francis, 2014.

Appendix A
Architecture of the Attack Model

TABLE XIV: Attack model layer sizes

Name	**Layers**	**Details**
Output Component	2 Fully Connected Layers	Sizes: 128, 64 Activation: ReLU Dropout: 0.2
Label Component	2 Fully Connected Layers	Sizes: 128, 64 Activation: ReLU Dropout: 0.2
Loss Component	2 Fully Connected Layers	Sizes: 128, 64 Activation: ReLU Dropout: 0.2
Gradient Component	Convolutional Layer	Kernels: 1000 Kernel size: $1\times$ Next layer Stride:1 Dropout: 0.2
	2 Fully Connected Layers	Sizes: 128, 64 Activation: ReLU Dropout: 0.2
Encoder Component	4 Fully Connected Layers	Sizes: 256, 128, 64, 1 Activation: ReLU Dropout: 0.2
Decoder Component	2 Fully Connected Layers	Sizes: 64, 4 Activation: ReLU Dropout: 0.2