Gradient-Leaks: Enabling Black-Box Membership Inference Attacks Against Machine Learning Models

Gaoyang Liu, *Member, IEEE*, Tianlong Xu, *Student Member, IEEE*, Rui Zhang, *Member, IEEE*, Zixiong Wang, *Student Member, IEEE*, Chen Wang, *Senior Member, IEEE*, and Ling Liu, *Fellow, IEEE*

***Abstract*— Machine Learning (ML) techniques have been applied to many real-world applications to perform a wide range of tasks. In practice, ML models are typically deployed as the black-box APIs to protect the model owner's benefits and/or defend against various privacy attacks. In this paper, we present Gradient-Leaks as the first evidence showcasing the possibility of performing membership inference attacks (MIAs), *with mere black-box access*, which aim to determine whether a data record was utilized to train a given target ML model or not. The key idea of Gradient-Leaks is to construct a local ML model around the given record which locally approximates the target model's prediction behavior. By extracting the membership information of the given record from the gradient of the substituted local model using an intentionally modified autoencoder, Gradient-Leaks can thus breach the membership privacy of the target model's training data in an unsupervised manner, without any priori knowledge about the target model's internals or its training data. Extensive experiments on different types of ML models with real-world datasets have shown that Gradient-Leaks can achieve a better performance compared with state-of-the-art attacks.**

***Index Terms*— Machine learning (ML), membership inference attack (MIA), black-box model, autoencoder.**

Manuscript received 25 February 2023; revised 26 July 2023 and 16 September 2023; accepted 4 October 2023. Date of publication 16 October 2023; date of current version 22 November 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62272183, Grant 52031009, Grant 62171189, and Grant U20A20181; in part by the Key Research and Development Program of Hubei Province under Grant 2023BAB074 and Grant 2021BAA026; in part by the Special Fund for Wuhan Yellow Crane Talents (Excellent Young Scholar); in part by Georgia Tech through USA NSF CISE under Grant 2302720, Grant 2312758, and Grant 2038029; in part by the IBM Faculty Award; and in part by CISCO Edge AI Program. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. George Theodorakopoulos. *(Corresponding author: Chen Wang.)*

Gaoyang Liu is with the Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: liugaoyang@hust.edu.cn).

Tianlong Xu, Zixiong Wang, and Chen Wang are with the Hubei Key Laboratory of Smart Internet Technology, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: tlxu@hust.edu.cn; zixwang@hust.edu.cn; chenwang@hust.edu.cn).

Rui Zhang is with the Hubei Key Laboratory of Transportation Internet of Things, School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China (e-mail: zhangrui@whut.edu.cn).

Ling Liu is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: ling.liu@cc.gatech.edu).

Digital Object Identifier 10.1109/TIFS.2023.3324772

I. Introduction

With the recent growth in computing power and improvement in algorithms, Machine Learning (ML) has become a core component of many real-world applications such as image recognition [1], natural language translation [2] and online advertising [3]. Driven by the success of ML, increasing companies and organizations integrate ML components into their services and products to improve the quality of service. However, recent works have demonstrated that ML models are vulnerable to various security and privacy attacks, such as the adversarial attacks [4], model stealing attacks [5], model inversion attacks [6], and privacy violation attacks [7].

In this paper, we concentrate on the so-called *membership inference attacks* (MIAs) against ML models, which aim to determine whether a data record (i.e., the target record) was used as part of the training dataset of a given ML model (i.e., the target model) [8], [9]. A successful MIA against an ML model signifies that the privacy of the training data was not sufficiently protected when the trained ML model is released. For instance, a research group releases an ML model which can predict the medicine dosage for a certain disease. By knowing that an individual's medical record was used to train this ML model, the attacker can thus infer that this individual is more likely to suffer from the corresponding disease.

Despite extensive research efforts on MIAs in recent years, most if not all existing works assume that the attacker has the prior knowledge of either the target ML model's internals (e.g., the model structure[1] [8], [10], parameters [11], [12], or the training loss [13]), or its training data (e.g., data samples [8], [14], [15], or data distribution [16]). In practice, developing an ML model is a product of massive costs and expertise efforts, including data collection, dataset annotation, model selection, parameter fine-tuning, etc. Therefore, the ML model is typically deployed as a black-box for protecting the model owner's benefits. For instance, machine learning as a service (MLaaS) platforms, including Google AI Platform,[2]

[1]MIAs specially designed for a particular ML model (e.g., deep learning models) are also regarded to have priori knowledge of the model structure.

[2]https://cloud.google.com/ai-platform

Amazon ML,[3] and BigML,[4] usually deploy their ML models as the black-box: neither the model owner nor the users can download the model; instead, they can only access the model through the provided black-box API. As a consequence, the applicability of existing MIAs is largely under restrictions, and it is so far unclear how to launch black-box MIAs for real-world attackers.

In this paper, we present Gradient-Leaks as the first evidence that it is possible to perform MIA *with mere black-box access*. Gradient-Leaks is based on the observation that each data record in the training set can influence the ML model's parameters in order to minimize its contribution to the model's training loss. The gradient of the loss on a target record with respect to this trained model thus indicates how much and in which direction the model needs to be changed to fit to the target record. When the training process completes, the trained model can fit to the whole training set with the smallest training loss. In this case, the training data's gradient will converge to a small magnitude in a certain direction. In contrast, the magnitude and direction of the gradient for a record that has not participated in the target model's training would be obviously different from those of a training record. As such, by leveraging the gradient difference between the data the target model trained on and that the target model meets for the first time, Gradient-Leaks can thus infer which record has been used to train the target model.

Although the basic idea sounds simple, Gradient-Leaks confronts two major challenges. The first challenge lies in the difficulty in obtaining the gradient of the target record directly, since we have no priori knowledge of either the target model or its training data, except for the black-box prediction access. Inspired by recent advances in explainable ML field [17], [18], [19], we propose to construct a local ML model around the target record which can locally approximate the target model's prediction behavior, and treat the target record's gradient with respect to the substituted local model as the gradient approximation of the target record.

The second challenge is that it is difficult to perform MIA well by directly leveraging the approximate gradient difference between the training data and the testing data, as the gradient of the target record mainly contains the explicit information about model optimization, while the membership information is usually hidden behind it [7]. To address this challenge, we intentionally modify the structure of an autoencoder to learn a representation for the approximate gradient, which is able to extract the implicit knowledge about the membership information. This representation is further utilized as features to distinguish the member from non-member records.

We summarize our major contributions as follows.

- We present Gradient-Leaks, an MIA against black-box ML models without requiring the priori knowledge about the target model or its training data, but only using the model's prediction interface.
- We show how a local linear ML model on the target record can be adopted to derive its gradient approximation to facilitate the black-box MIA.
- We propose to construct the inference attack model in an unsupervised manner, based on the extracted membership features from the gradient approximation using a modified autoencoder, which relaxes the assumption of information about the training data.
- We evaluate the performance of Gradient-Leaks against five different types of ML models (two of them instantiated by real-world MLaaS platforms), and compare with three representative MIAs on four realistic datasets. The results show that Gradient-Leaks performs better than the state of the arts, even with mere black-box access.

The remainder of this paper is organized as follows. Section II overviews related works on MIAs. Section III presents the threat model. Section IV describes the design of Gradient-Leaks, followed by the performance evaluation in Section V. Finally, Section VI concludes this paper. The code of Gradient-Leaks has been released for reproducibility purposes.[5]

II. Related Work

In this section, we briefly overview recent advances in MIAs, according to different priori knowledge required by the attacker (c.f. Table I). Generally, there are two categories of assumptions on the required knowledge: regarding the target model (including the model structure, hyper-parameters, and training loss), and its training data (including the data distribution, and part/all of the training samples).

A. Attacks With Both Target Model and Training Data

Shokri et al. [8] present the first MIA dubbed Shadow Attack against ML models. They construct multiple shadow models to mimic the prediction behavior of the target model, and the shadow models' outputs are further used to train the attack model. This attack requires the priori knowledge about the target model's structure and its training data's distribution (or a part of its training data).[6] In ML-Leaks [14], Salem et al. extend Shadow Attack and show that it is possible to perform the same attack with only one shadow model instead of multiple shadow models. They still need the target model's structure and part of the training samples. Recently, Li et al. [15] propose an instance-probability attack with multiple shadow models trained on the data that has the same distribution as that of the target model's training data. They use the prediction probability of these shadow models to extract the membership feature to perform the MIAs. Song and Mittal [20] propose M-Entropy Attack, where they develop

[3]https://aws.amazon.com/machine-learning

[4]https://bigml.com

[5]https://www.dropbox.com/s/pi2xdzlow8as36s/Gradient-Leaks-Code.zip?dl=0

[6]As for the model structure knowledge of the target model, when attacking MLaaS platforms, although Shokri et al. cannot explicitly get the structure of the target model, they can get the shadow models with the same structure as the target model by leveraging the same MLaaS platform. In addition, according to their released code (https://github.com/csong27/membership-inference), when attacking NN models, Shokri et al. directly use the target model's structure to train the shadow models. Therefore, we also treat this scenario as the attacker with the structural information of the target model.

TABLE I
SUMMARY OF EXISTING MEMBERSHIP INFERENCE ATTACKS

Existing Attacks	Priori Knowl. of Target Model			Priori Knowl. of Training Data	
	Structure	Parameters	Training Loss	Samples	Distribution
Shadow Attack [8]	✓	⊗	⊗	✓	✓
ML-Leaks (Type 1) [14]	✓	⊗	⊗	⊗	✓
Li et al. [15]	✓	⊗	⊗	⊗	✓
Wu et al. [11]	N/A	✓	⊗	✓	⊗
Yeom et al. [13]	⊗	⊗	✓	✓	⊗
M-Entropy Attack [20]	✓	⊗	⊗	✓	✓
TrajectoryMIA [21]	✓	⊗	✓	⊗	✓
Label-Only MIA [22]	✓	⊗	⊗	⊗	✓
Nasr et al. (Type 1) [12]	✓	✓	⊗	⊗	⊗
Nasr et al. (Type 2) [12]	✓	⊗	⊗	⊗	⊗
Hayes et al. [23]	✓	✓	⊗	⊗	⊗
Liu et al. [24]	✓	✓	⊗	⊗	⊗
Melis et al. [25]	✓	✓	⊗	⊗	⊗
Sablayrolles et al. [10]	⊗	✓	✓	⊗	⊗
Truex et al. [26]	⊗	⊗	⊗	✓	⊗
Liu et al. [16]	⊗	⊗	⊗	⊗	✓
ML-Leaks (Type 2) [14]	⊗	⊗	⊗	⊗	✓
LiRA [27]	N/A	⊗	⊗	⊗	✓
L-Leaks [28]	⊗	⊗	⊗	⊗	✓
Transfer Attack [29]	⊗	⊗	⊗	⊗	✓
Gradient-Leaks	⊗	⊗	⊗	⊗	⊗

Required Knowledge: ✓ Non-required Knowledge: ⊗ N/A: Not Available

a modified prediction entropy metric that incorporates the ground truth label of the target record. They then determine distinct threshold values for each class label, which are learned using the shadow training technique employed in Shadow Attack [8]. Subsequently, they classify the target record as a member if its modified entropy is below the preset threshold. Choquette-Choo et al. [22] consider a more restricted scenario in which the target model only returns the predicted labels and propose a label-only MIA. Their MIA first trains a shadow model on a dataset drawn from the same distribution of the target model's training set and then utilize adversarial attacks to estimate the record distance to the decision boundary of the shadow model to perform the attack.

Except for leveraging shadow models, some researchers use other information of the target model to launch MIAs. Wu et al. [11] use the target model's parameters and an auxiliary dataset to compute the membership probability, and then select a threshold to obtain the attack model. This auxiliary dataset consists of samples from the training/testing datasets and the ground truth of each sample's membership property. Yeom et al. [13] propose a simple MIA by leveraging the training loss of the training data. They first query the target model with all training data, and obtain the average training loss which serves as the threshold for MIA. This work requires all training data and the corresponding training loss.

Liu et al. [21] exploit the membership information from the training process of the target model and design a new MIA, called TrajectoryMIA. They use the knowledge distillation technique to mimic the training process of the target model, and then extract the membership information from the loss of the target records on the intermediate models at different distillation epochs with the loss from the given target model.

B. Attacks With Mere Target Model

Since requiring information on both the target model and its training data is impractical to some extent, many studies concentrate on relaxing such assumptions, and seek for techniques given mere the target model's internals. Nasr et al. [12] present two types of MIAs against deep learning models by exploiting the MIA vulnerabilities of the stochastic gradient descent (SGD) algorithm. For training the attack model, one MIA employs the internal computation results of the target model (including the activations and gradients on the target data record), and the other leverages the activation outputs of individual layers of the target model. To perform such attacks, they need the knowledge about the target model's internal structure and parameters. Melis et al. [25] also design an MIA for deep learning models using the internal computation results (i.e., the gradients of the embedding layers). The non-zero gradients of the embedding layers reveal which features appear in a batch and are used to infer the record membership, but require the parameters of the target model's embedding layers.

In addition, some studies focus on performing MIAs on generative models. Hayes et al. [23] present an MIA against generative models. They create a local copy of the discriminator of the target model, and then leverage the prediction of the copy model on the target record to perform the attack.

This MIA requires the inner parameters of the target model. With white-box access, Liu et al. [24] propose an MIA named co-membership attack against generative models. They optimize an attacker network to search for the latent encoding to reproduce the target record, and then the reconstruction error is used directly to infer the membership.

Although many MIAs have been proposed, there is still a lack of theoretical analysis. Sablayrolles et al. [10] recently leverage a probabilistic framework to derive a formal analysis for their optimal MIA attack, which relies on either the target model's training loss or its parameters.

C. Attacks With Mere Training Data

Some studies also attempt to design MIAs with mere training data information. Truex et al. [26] demonstrate how Shadow Attack can be leveraged in adversarial ML settings. They assume that the attacker has samples of his own which can be used as seeds for shadow data generation, and expose MIA vulnerabilities through the perspectives of data skewness and adversarial learning. Instead of constructing the shadow model, Liu et al. [16] train a mimic ML model based on a synthetic dataset with the same distribution of the training data. Without any knowledge about the target model, they propose an imitation method based on the generative adversary networks to mimic the prediction behavior of the target model to perform MIAs. In ML-Leaks [14], Salem et al. also propose to construct a set of ML models using data with the same distribution of the target model's training data, each with a different classification algorithm, and combine these models together as one shadow model to launch MIAs. Li and Zhang [29] propose Transfer Attack which leverages the intuition that the transferability property holds between the shadow model and the target model. They input the target record into the shadow model and calculate its loss with the ground truth label, and then determine the record is a member if the loss value is smaller than a threshold.

Since the logits of the target model provide more prediction information, Yan et al. [28] present L-Leaks which could approximate the logits of the black-box target model to improve the similarity between the substitute and target models. Then L-Leaks allows the attacker to use the substitute model's information to perform MIAs.

Except for constructing substitute models, Carlini et al. [27] develop a Likelihood Ratio Attack, which trains several models on the data that has the same distribution as that of the target model's training data. Then they estimate the prediction distributions of models trained with and without a certain data record, where MIA is performed by comparing the prediction of the target model with the estimated distributions.

Summary: It is observed that existing MIAs require pre-knowledge of either the target model's internals or its training data. In practice, however, the model owner often only provides the black-box prediction API to users. It is therefore unclear if MIAs can be launched with only black-box information. Against this background, we put forward to leverage the gradient approximation to extract the membership features to facilitate unsupervised MIA, without knowing any prior information but the mere black-box API, thereby revealing the possibility of black-box membership leakage.

III. Threat Model

We consider an adversary who seeks to determine whether or not a data record was used to train the target ML model. The adversary is assumed to have only the black-box access to the target model. Thus the adversary has no access to the target model internals or its training data, and can only query the target model with a data record to obtain the corresponding prediction probability vector. The details of our threat model are described as follows.

A. Target Model

We focus on attacks targeting the classification models, regardless of what type of the model is. When querying the target model $\mathcal{M}$ with a data record, it will output the prediction result where each value represents the probability that the input record belongs to the corresponding class. We formalize the target model as $\mathcal{M} : \mathbf{x} \rightarrow y$, where $\mathbf{x}$ is the input data and y is the corresponding predicted probability. Since the adversary only has the black-box access to the target model, $\mathcal{M}$ represents the prediction interface of the target model.

B. Adversary Priori Knowledge

In order to perform our MIA, we consider a weak (but more practical) adversary with mere black-box access to the target model. Such a case is referred to as the black-box setting, where the adversary cannot obtain any priori knowledge about the following information:

- *Target model structure:* including the type and the structure of the target model.
- *Target model parameters:* including the internal parameters, hyper-parameters such as regularization parameters, and the number of epochs used to train the target model.
- *Target model training process:* including the training process of the target model, and the training loss therein.
- *Auxiliary dataset or distribution of the training data:* including any dataset that shares the same distribution as the training set used to create the target model, as well as the distribution of the training data. In this sense, it is almost infeasible to obtain a dataset that can be used to train a substitute model to imitate the prediction behavior of the target model.
- *Membership information of the training data:* Given a dataset containing multiple target samples, we don't have any information about the membership property of each individual sample. Specifically, what we could know is that a part of the samples is in the training dataset. However, the exact membership property of individual samples within the given dataset still remains unknown to the attacker.

C. Adversary Capability

In our black-box settings, the only capability of the adversary is to obtain the prediction result by querying $\mathcal{M}$

with a record $\mathbf{x}$:

$$\mathcal{M}(\mathbf{x}) = [y^1, y^2, \cdots, y^c, \cdots, y^{|C|}] \tag{1}$$

where the prediction result is a probability vector and C is the set of class labels that $\mathcal{M}$ can take. Each value y^c ($c \in C$) in this vector corresponds to the predicted probability that this class c is the correct label.

D. Adversary Goal

Given the target model $\mathcal{M}$ and the target record $\mathbf{x}_t$, the adversary goal is to infer whether $\mathbf{x}_t$ is in $\mathcal{M}$'s training set or not:

$$\mathcal{A}(\mathbf{x}_t, \mathcal{M}) \rightarrow \textbf{In/Out} \tag{2}$$

where A represents the attacking functionality of Gradient-Leaks. The label **In** (resp. **Out**) represents that the adversary believes $\mathbf{x}_t$ belongs to the target model's training set (resp. testing set).

IV. Design of Gradient-Leaks

We formalize the task of Gradient-Leaks as follows: given the black-box access to the target ML model $\mathcal{M}$, Gradient-Leaks first approximates the local gradient of a target record $\mathbf{x}_t$ on $\mathcal{M}$, then extracts the membership features from the gradient approximation, and finally determines whether $\mathbf{x}_t$ was used to train $\mathcal{M}$ or not leveraging the gradient approximation. To this end, Gradient-Leaks mainly involves the following three steps.

Gradient Approximation: In order to approximate the local gradients of $\mathcal{M}$, we first sample a set of data records around $\mathbf{x}_t$ by perturbing the feature values of $\mathbf{x}_t$, and then query the target model with these sampled data to obtain the corresponding prediction results. With these results, a local linear regression model around the target record $\mathbf{x}_t$ can thus be constructed, and the gradients of the local model with respect to $\mathbf{x}_t$ can be regarded as the gradient approximation of $\mathcal{M}$ (c.f. Fig. 1).

Membership Feature Extraction: We next modify an autoencoder to extract the membership features from $\mathbf{x}_t$'s approximate gradient. Specifically, we replace the input and output of the standard autoencoder with the approximate gradient and several signals related to the membership property, respectively (c.f. Fig. 2). Then the latent embedding of the trained autoencoder serves as the membership features for $\mathbf{x}_t$.

Membership Inference: Given the membership features of a group of records, an unsupervised clustering algorithm can be then leveraged to construct an attack model which can cluster these records in two clusters, ultimately separating members from non-members.

A. Gradient Approximation

The key idea of the gradient approximation is to construct a locally faithful model, whose parameter gradients can be derived easily, to mimic the target model's prediction behavior around the target record. For the sake of simplicity, we choose linear regression to construct our local model. The local model's prediction accuracy is closely related to the *local fidelity* [30], which gives us an idea of how well

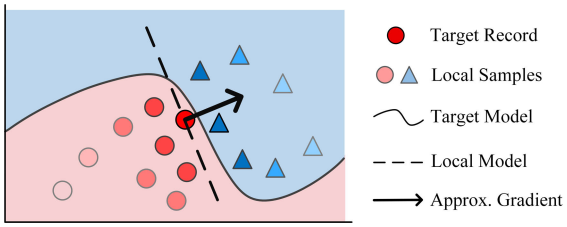


Fig. 1. A toy example of the gradient approximation, where the color intensity of the local sample represents its contribution to the local model. A linear model is constructed with the vicinal samples around the target record to approximate the target model locally, and its gradient is regarded as the gradient approximation of the target model.

the local model approximates the target model's predictions around the target record. If the local model can achieve a resembling prediction performance compared with the target model around the target record, it should be locally faithful and can be used to estimate the gradient of the target record.

Formally, we construct a local linear regression model, denoted as $\mathcal{R}$, to approximate the prediction behavior of the target model in the vicinity of the target record. We use $\pi_{\mathrm{x}}(\tilde{\mathbf{x}}_l)$ as a similarity measure between the target record $\mathbf{x}_t$ and another record $\tilde{\mathbf{x}}_l$, so as to define the locality around $\mathbf{x}_t$. Then the local model $\mathcal{R}$ is obtained by the following loss function:

$$\arg\min_{\mathcal{R}} \quad \mathcal{L}(\mathcal{M}, \mathcal{R}, \pi_{\mathrm{x}}) \tag{3}$$

where we denote $\mathcal{L}(\mathcal{M}, \mathcal{R}, \pi_{\mathrm{x}})$ as a measurement of how faithful $\mathcal{R}$ is in approximating $\mathcal{M}$ around the target record. The gradients of $\mathcal{R}$ can thus serve as the approximation results of $\mathcal{M}$'s gradients over $\mathbf{x}_t$.

1) Local Sample Generation: The first step of the gradient approximation is to generate a set of data samples around $\mathbf{x}_t$ with corresponding sample weight π_{x}. Since we do not have any priori knowledge about the target model's training data, we perform the data generation by leveraging the perturbations of $\mathbf{x}_t$. We select a set of features of $\mathbf{x}_t$ that will be perturbed uniformly at random, and denote the selected feature set as F_p. Given a target record $\mathbf{x}_t$, we perturb a part of $\mathbf{x}_t$'s feature values according to the selected features in F_p. Specifically, we successively replace the selected feature value of $\mathbf{x}_t$ with a different value which is randomly chosen from the value range of the corresponding feature. Then we use this perturbed record as the generated sample around $\mathbf{x}_t$, which is denoted as $\tilde{\mathbf{x}}_l$. One thing should be noted that the number of perturbed features $N_{feature}$ is not a constant number. It is randomly chosen at the beginning of the generation of each sample, so $N_{feature}$ can vary within the total number of the target record's features. Using our feature perturbation-based sample generation method, we can generate one local sample at a time, which can allow us to get any desired number of samples.

Naturally, the perturbed sample $\tilde{\mathbf{x}}_l$ can be in the vicinity of $\mathbf{x}_t$ or far away from $\mathbf{x}_t$. In order to guarantee our local model's prediction accuracy in the neighborhood of the target data, we force the sample which is far away from the target record with a smaller contribution to the local model through defining a sample weight for $\tilde{\mathbf{x}}_l$. As for calculating the weight

of $\tilde{\mathbf{x}}_l$, we use an exponential function as follows:

$$\pi_{\mathrm{x}}(\tilde{\mathbf{x}}_l) = \exp(-Dist(\mathbf{x}_t, \tilde{\mathbf{x}}_l)) \tag{4}$$

where $Dist$ represents a distance function such as cosine distance, $L2$ distance, and Hamming distance. We repeat the above steps to generate local samples around $\mathbf{x}_t$ until the samples are enough. For clarity, we denote all generated samples as a dataset D_{local}, and the sample number of D_{local} as N_{local}. The weights of the perturbed samples can help our local model achieve a resembling prediction performance with the target model in the neighborhood of the target data and get a high local fidelity.

2) Local Model Construction: Now that we have got a dataset D_{local} which is sampled around the target record $\mathbf{x}_t$ and weighted by π_{x}. Next, Gradient-Leaks needs to build an ML model that has a similar prediction behavior with the target model locally around $\mathbf{x}_t$.

Specifically, we first leverage $\mathcal{M}$'s prediction interface to get each sampled record's prediction results which represent the probabilities that the records of $\tilde{\mathbf{x}}_l$ belong to each class, and we denote the results as $y_{\tilde{\mathbf{x}}_l}$. Then for each class $c \in C$, we train a local linear model $\mathcal{R}_c$ around the target record $\mathbf{x}_t$ by minimizing the following loss function:

$$\begin{aligned}\mathcal{L}_c(\mathcal{M}, \mathcal{R}_c, \pi_{\mathrm{x}}) &= \frac{1}{2} \sum_{\tilde{\mathbf{x}}_l \in D_{local}} \pi_{\mathrm{x}}(\mathcal{R}_c(\tilde{\mathbf{x}}_l) - \mathcal{M}(\tilde{\mathbf{x}}_l)|_c)^2 \\ &= \frac{1}{2} \sum_{\tilde{\mathbf{x}}_l \in D_{local}} \pi_{\mathrm{x}}(\tilde{\mathbf{x}}_l)(w_c^T \tilde{\mathbf{x}}_l + b_c - y^c_{\tilde{\mathbf{x}}_l})^2 \end{aligned} \tag{5}$$

where $\mathcal{R}_c(\mathrm{x}) = w_c^T \mathrm{x} + b_c$. $\mathcal{M}(\tilde{\mathbf{x}}_l)|_c$ and $y^c_{\tilde{\mathbf{x}}_l}$ both represent the probability that $\tilde{\mathbf{x}}_l$ belongs to class c predicted by $\mathcal{M}$. Here we use the mean squared error (MSE) loss as $\mathcal{L}$ for constructing $\mathcal{R}_c$. Thus $\mathcal{L}_c$ is the loss value of the local model $\mathcal{R}_c$. At the end of this step, we get a set of local linear models corresponding to different classes: $\mathcal{R} = [\mathcal{R}_1, \mathcal{R}_2 \cdots \mathcal{R}_{|C|}]$. Then we leverage these local linear models to derive the approximate gradients of $\mathcal{M}$ with respect to $\mathbf{x}_t$.

In Gradient-Leaks, the quantity of local samples utilized for training the local model significantly influences the attack performance (c.f. Fig. 6). Therefore, we need to determine the number of local samples in advance. Since our generation method allows us to produce local samples incrementally, we can gradually increase the number of local samples and train the local models until the prediction discrepancy between the local model and the target model becomes smaller without significant variations. Then we use the current number of the entire local samples as the default value of our attack.

3) Local Gradient Approximation: Since the model $\mathcal{R}_c$ is locally faithful with the target model $\mathcal{M}$ on prediction behavior of class c, we leverage the gradients of all local models in the set $\mathcal{R}$ as approximation results of $\mathcal{M}$ over $\mathbf{x}_t$. Therefore, we successively calculate the parameter gradients of every model in $\mathcal{R}$ as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}_c}{\partial w_c} &= -(w_c^T \mathbf{x}_t + b_c - y^c_{\mathrm{x}})\mathbf{x}_t \\ \frac{\partial \mathcal{L}_c}{\partial b_c} &= -(w_c^T \mathbf{x}_t + b_c - y^c_{\mathrm{x}}) \end{aligned} \tag{6}$$

Algorithm 1 Gradient Approximation

Require: Target model $\mathcal{M}$; Weight function π_{x}
Require: Target record $\mathbf{x}_t$; Number of samples N_{local}
Require: Class set C; Local loss function $\mathcal{L}$
  $D_{local} \leftarrow \{\}$, $\nabla \boldsymbol{W} \leftarrow \{\}$, $\nabla \boldsymbol{b} \leftarrow \{\}$
  for $i \in \{1, 2, 3, \cdots, N\}$ **do** ▷ Generate local samples
    $\tilde{\mathbf{x}}_l \leftarrow sample\_around(\mathbf{x}_t)$
    $D_{local} \leftarrow D_{local} \cup \langle \tilde{\mathbf{x}}_l, \pi_{\mathrm{x}}(\tilde{\mathbf{x}}_l) \rangle$
  end for
  for $c \in C$ **do**
    $\mathcal{R}_c \leftarrow \mathbf{fit}(D_{local}, \mathcal{M}(D_{local})|_c)$ ▷ Fit the local model
    $\mathcal{L}_c \leftarrow \sum_{\tilde{\mathbf{x}}_l \in D_{local}} \pi_{\mathrm{x}}(\mathcal{R}_c(\tilde{\mathbf{x}}_l) - \mathcal{M}(\tilde{\mathbf{x}}_l)|_c)^2$
    $\nabla \boldsymbol{W} \leftarrow \nabla \boldsymbol{W} \cup \frac{\partial \mathcal{L}_c}{\partial W_c}$ ▷ Compute gradient with Equ. (6)
    $\nabla \boldsymbol{b} \leftarrow \nabla \boldsymbol{b} \cup \frac{\partial \mathcal{L}_c}{\partial b_c}$
  end for
  return $\nabla \boldsymbol{W}, \nabla \boldsymbol{b}$

where w_c and b_c are the parameters of the local model $\mathcal{R}$, $\frac{\partial \mathcal{L}_c}{\partial w_c}$ and $\frac{\partial \mathcal{L}_c}{\partial b_c}$ are the gradients of $\mathcal{R}_c$ over $\mathbf{x}_t$, and y^c_{x} represents the probability obtained from the target model that $\mathbf{x}_t$ belongs to class c.

Finally, the approximate gradients of the target model $\nabla \boldsymbol{W} = [\frac{\partial \mathcal{L}_1}{\partial w_1}, \frac{\partial \mathcal{L}_2}{\partial w_2} \cdots \frac{\partial \mathcal{L}_C}{\partial w_C}]$ and $\nabla \boldsymbol{b} = [\frac{\partial \mathcal{L}_1}{\partial b_1}, \frac{\partial \mathcal{L}_2}{\partial b_2} \cdots \frac{\partial \mathcal{L}_C}{\partial b_C}]$ are obtained, which are further exploited to infer whether $\mathbf{x}_t$ is in the target model's training set or not. The gradient approximation algorithm is outlined in Algorithm 1.

B. Membership Feature Extraction

Now that we have got the approximate gradients $\nabla \boldsymbol{W}$ and $\nabla \boldsymbol{b}$ of the target model $\mathcal{M}$, we next exploit these gradient information to infer the membership information about $\mathcal{M}$'s training set. The most straightforward method for membership inference is to construct a binary attack model to determine the membership property of a given record according to its approximate gradients. However, the membership information about a given record is usually hidden behind its approximate gradients, so we need to extract the membership information.

To extract the membership features for each data record, we leverage an autoencoder to achieve this purpose. An autoencoder is a neural network that learns efficient data representation in an unsupervised manner [31], [32]. Internally, it has a hidden layer that describes a representation used to represent the input. The typical structure of an autoencoder consists of two main parts: an *Encoder* that reduces the input dimensions and compresses the input data into an encoded representation, and a *Decoder* that reconstructs the data from the encoded representation to be as close to the original input as possible. The autoencoder is trained to minimize the reconstruction errors as follows:

$$\mathcal{L}(\mathrm{x}, \hat{\mathrm{x}}) = \|\mathrm{x} - \hat{\mathrm{x}}\|^2 \tag{7}$$

where x is the input data and $\hat{\mathrm{x}}$ is the reconstruction output.

The standard autoencoder, however, brings no extra benefit for Gradient-Leaks to learn a data representation, since the membership information of a given record is still hidden

behind the learned representation. Fortunately, it is shown that by adding extra regularization [33] or loss term [34] to the reconstruction loss, the autoencoder is able to generate output data which resembles the input data, rather than just duplicating the input data. Moreover, the autoencoder can even be forced to prioritize which aspects of the input should be copied and learn a low-dimension representation which contains a part of the properties of the input data [35], [36]. Thus in our design, we intentionally modify the input and output of the standard autoencoder (c.f. Fig. 2), so as to learn a low-dimension representation of the input data which can reflect the membership property.

According to existing works [8], [12], we find some signals that can be leveraged in our autoencoder. As discussed in [8], attacking with only the prediction probability of the top 1 class can achieve a resembling attack performance compared with that using the whole probability vector. Therefore, the probability of the most likely class predicted by the target model can reflect the membership property of the corresponding record. In addition, disturbing the uncertainty of the prediction vector can mitigate the risk of MIAs. That is because if the target record was used to train the target model, the target model would have high confidence to predict this record belongs to a certain class and the prediction uncertainty will be close to 0 in such a case. Thus the original uncertainty of the prediction vector is another signal that can be exploited by our autoencoder. For the prediction uncertainty, we use the normalized entropy of the prediction probability vector for a given record as follows:

$$H(\mathcal{R}(\mathrm{x})) = \frac{1}{\log(|C|)} \sum_{c=1}^{|C|} \hat{y}_{\mathrm{x}}^{c} \log(\hat{y}_{\mathrm{x}}^{c}) \tag{8}$$

where $\hat{y}_{\mathrm{x}}^{c}$ represents the probability that the target record belongs to class c predicted by the local model $\mathcal{R}$.

Moreover, as discussed in [12], the gradients of the target model's loss over the training data are statistically smaller than those over the testing data, as the objective of ML algorithms is to minimize the training data's loss with respect to the ML model. Therefore, the norm of our approximate gradient also contains the membership information. As for the gradient norm, we make use of the $L2$ norm to measure the magnitude of our approximate gradient (i.e., $|\nabla \boldsymbol{W}|$ and $|\nabla \boldsymbol{b}|$).

In our modified autoencoder, we feed the *Encoder* with the approximate gradients $\nabla \boldsymbol{W}$ and $\nabla \boldsymbol{b}$, and the output of the *Decoder* includes the prediction uncertainty of the local model $H(\mathcal{R}(\mathrm{x}))$, the norm of the gradients $\|\nabla \boldsymbol{W}\|$ and $\|\nabla \boldsymbol{b}\|$, and the probability of the prediction result on the correct label (the label predicted by the target model) $\mathcal{R}(\mathrm{x})_{y=\mathcal{M}(\mathrm{x})}$ accordingly. During the training process of the modified autoencoder, the *Encoder* tries to extract the information that the approximate gradients contain regarding these features.

After training the autoencoder, the *Encoder* can generate the latent embedding z for the target record in a low-dimensional space, which contains the membership information and thus can help Gradient-Leaks easily distinguish the member from non-member records. Then we leverage the *Encoder* output z as the membership features to facilitate our membership inference (c.f. Fig. 2).

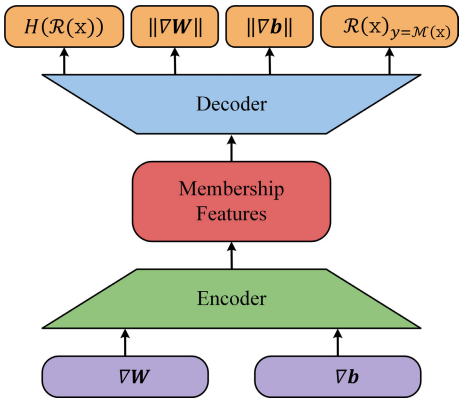


Fig. 2. A schematic view of the membership feature extraction.

It should be noted that Nasr et al. [12] use an autoencoder to directly predict a membership score for each data record, representing the membership probability of the corresponding input record. However, compressing the gradient information of the record to a single value will bring relatively large information loss and make it impossible to highlight the membership information. To address this issue, we increase the size of the autoencoder's bottleneck from 1 to a larger value to expand the representation capacity of the encoder's output. Then we treat the output of the encoder part as the embedding of membership information of the target samples to perform our MIA.

C. Membership Inference

The goal of Gradient-Leaks is decisional membership inference, thus we construct an attack model that is a binary classifier with two output classes, **In** and **Out**. Since we have no access to the target model's training data, we cannot obtain the ground truth of the given target record whether it is used to train the target model or not. Therefore, the commonly adopted supervised MIA methods are no longer applicable in our settings, and we choose to utilize the unsupervised approach to perform our membership inference.

Given a dataset D' consisting of multiple target records we suspect in $\mathcal{M}$'s training set, Gradient-Leaks attempts to infer the membership property of all target records simultaneously, in an unsupervised manner. Now that we have obtained the membership features of the records in D', we can simply leverage an unsupervised clustering algorithm to construct the attack model. It is natural to cluster the records of D' into two clusters and then determine the cluster with a lower mean gradient norm as the members of the target model's training set. However, through our experiments, we find that Gradient-Leaks may perform better when the cluster number is larger. Therefore, we involve a multilevel clustering method to infer the membership property of the target records. We first cluster the target samples into a larger number of clusters, and then group the cluster centroids into two clusters. With our multilevel clustering method, we could handle the data samples whose membership features do not fall exactly into both membership and non-membership clusters. It is worth noting

that any classic clustering algorithm (including K-Means [37], DBSCAN [38], and spectral clustering [39]) can be employed.

During the training process, the attack model attempts to find similarities in membership features of the records in D' and group similar training records together. In the end, the attack model separates the records into two clusters, where the cluster with a smaller mean norm of gradients is labeled as the class **In**, and the other as **Out**.

V. Performance Evaluation

A. Experiment Setup

1) Datasets: We utilize four different datasets to evaluate the performance of Gradient-Leaks. Among them, three datasets including UCI Adult,[7] Purchase,[8] and MNIST[9] are the same as those used in the previous MIAs [8], [12], [14]. We also make use of Bank Marketing dataset,[10] which is obtained from the financial field.

a) UCI adult (census income): This dataset includes 48, 842 records with 14 attributes such as age, gender, education, marital status, occupation, working hours, and native country. The classification task of this dataset is to predict if a person earns over \$50K a year based on the census attributes.

b) Purchase: Purchase dataset contains shopping histories of several thousand shoppers over one year, including many fields such as product name, store chain, quantity, and date of purchase. In particular, Purchase dataset (with 197, 324 records) does not contain any class labels. Following Shokri et al. [8] and Salem et al. [14], we adopt K-Means algorithm to assign each data record with a class label. The numbers of classes include 2, 10, 20, 50, and 100, and each class corresponds to a purchase style.[11]

c) MNIST : This is a dataset of 70,000 handwritten digits formatted as 32×32 images and normalized so that the digits are located at the central of the image. It includes sample images of handwritten digits from 0 to 9. Each pixel within the image is represented by 0 or 1.

d) Bank marketing : This dataset includes 45, 211 client information of a Portuguese banking institution, and the goal is to predict if the client will subscribe to a term deposit (binary classification). This dataset contains 17 attributes such as marital, education, personal loan, and type of job.

For each dataset, 10,000 records are randomly selected to train different types of target models.

2) Target Models: We evaluate Gradient-Leaks on five different types of target models: three implemented locally and two constructed by the cloud-based MLaaS platforms. In our experiments, we treat all the target models as black boxes.

a) Native target models : We locally construct three types of ML models as the target models, including logistic regression (LR), random forest (RF), and deep neural network (DNN). We use the standard training process provided by the ML software libraries scikit-learn [40] (for LR and RF) and PyTorch [41] (for DNN) to build these target models. After the training process, we only provide the prediction interface of the target models to Gradient-Leaks to perform the attacks.

b) Cloud-based target models: In our experiments, the cloud-based target models are trained by two MLaaS platforms. The first platform is Amazon ML, where the user cannot choose the model types but can modify a few parameters, including the maximum number of passes over the training data and $L2$ regularization amount. The former determines the number of training epochs and the latter tunes how much regularization is performed on the model parameters in order to avoid overfitting. We use Amazon ML platform to train the target models with the same parameter setting, in which the number of epochs is 200, and the $L2$ norm is 10^{-6}.

The second cloud-based MLaaS platform is BigML. Different from Amazon ML platform, the user of BigML is allowed to select the model's type and manipulate the model's parameters. However, we do not participate in the training process of the target models on BigML platform. In our experiments, the type of the target models is chosen by BigML adaptively depending on the data, and all the parameters of the target models are set to the default values.

3) Evaluation Metrics: We evaluate the performance of Gradient-Leaks using ***precision*** and ***recall*** metrics of MIAs. Specifically, ***precision*** presents the proportion of the data records predicted as members of the training dataset that are indeed in the target model's training set. ***Recall*** presents the fraction of the training records that we can correctly infer as the training set's records. In other words, ***precision*** (resp. ***recall***) measures the accuracy (resp. coverage) of MIAs.

Furthermore, in order to evaluate how close the prediction behavior of our local model is with the target model around the given record (i.e., local fidelity), we adopt the following two metrics. One is the ***local accuracy*** of the local linear model, which is defined as the ratio of the size of the local samples predicted to the same class by both the local model and target model to the size of all local samples. The other metric is ***L1 Norm*** of the prediction probability difference between the local linear model and the target model on the same target record, which can be obtained by:

$$P_{diff} = \|y_x - \hat{y}_x\|_1 \tag{9}$$

where y_x (resp. $\hat{y}_x$) is the prediction result of the same target record obtained from the target (resp. local) model.

4) Comparison Methods: We consider the following state-of-the-art MIAs as our comparison methods:

a) Shadow attack [8]: It builds multiple shadow models with the same structure as the target model to mimic the target model's prediction behavior, and leverages the shadow model's outputs to train an attack model that can separate the member and non-member of the target model's training set. In our experiments, we set the number of shadow models used by Shadow Attack to 10.

b) ML-Leaks [14]: It constructs a set of sub-shadow models with different algorithms, and combines them together as one shadow model to generate the training data for obtaining the inference attack model. In keeping with

[7] https://archive.ics.uci.edu/ml/datasets/Adult

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

[9] http://yann.lecun.com/exdb/mnist/

[10] https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

[11] Unless otherwise specified, the dataset Purchase in the subsequent sections of this paper specifically refers to the dataset with 100 classes.

the experiment settings of ML-Leaks, we construct three sub-shadow models using RF, LR, and NN algorithms and stack them together as one shadow model.

c) Output attack [12]: It builds an unsupervised attack model by using the target model's outputs, and predicts the cluster with a lower uncertainty as the member of the training set.

d) M-Entropy attack [20]: It calculates the entropy by considering both the prediction probability of the correct label and the entropy of the prediction probabilities for other incorrect labels. Based on the modified entropy, the target record is classified as a member if it falls below the predetermined threshold, and as a non-member otherwise. The thresholds are set for each class individually, which are learned using the shadow training technique [8].

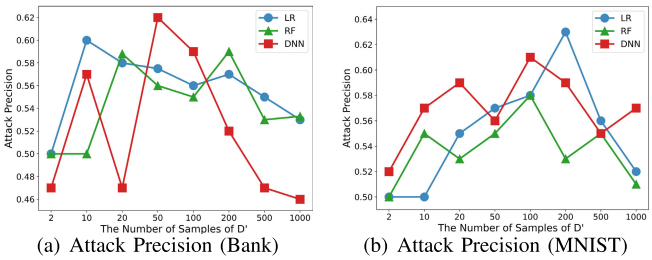
We perform all the above attacks as well as Gradient-Leaks on randomly reshuffled records from the target model's training and testing datasets, where the number of members is set equal to the number of non-members, in order to maximize the uncertainty of inference (thus the baseline accuracy is 0.5, equivalent to random guess). When it comes to Gradient-Leaks, we set the default number of local samples to 5,000. The architecture of the modified autoencoder in our attack comprises a 4-layer encoder and a 2-layer decoder. The hidden layer size, which represents the dimension of membership features, is set to 5. To optimize the model, we employ SGD with a training epoch of 1,000 and a learning rate of 10^{-3}.

B. Impact of Number of Suspicious Dataset

In order to evaluate the impacts of the record number of suspicious dataset D', we measure the attacking performance of Gradient-Leaks against three types of native target models, where the number of local samples is set to 5,000. From Fig. 3(a) we can see that for the target models trained on Bank dataset, Gradient-Leaks can achieve a mean attack precision of 0.595 with $|D'| = 50$. When D' contains only two target records, Gradient-Leaks can only achieve a mean precision of 0.505. As the number of D' increases to 500 (resp. 1,000), the attack precision gradually decreases by 6.8% (resp. 7.7%).

Regarding the target models trained on MNIST dataset, Gradient-Leaks can achieve a mean attack precision of 0.583 when $|D'| = 100$. As the number of D' increases to 200, Gradient-Leaks can still achieve a mean precision of 0.576 as shown in Fig. 3(b). However, when D' only contains two target records, Gradient-Leaks achieves a mean precision of 0.513 which is just slightly higher than that of random guess. When D' consists of 500 (resp. 1,000) records, the performance of Gradient-Leaks decreases by 3.6% (resp. 5.6%).

Similar results can also be observed when using Adult and Purchase datasets. The results demonstrate that the number of D' would largely affect the attack performance of Gradient-Leaks. Neither too small nor too large number of D' allows Gradient-Leaks to achieve a satisfactory inference performance, and this phenomenon is especially obvious when attacking against the DNN model. Referring to the above results, we thus empirically set the record number of D' to 100 in the following experiments, no matter which dataset the target model is trained on and which type of algorithm the target model employs.



(a) Attack Precision (Bank) (b) Attack Precision (MNIST)

Fig. 3. The impact of the record number in D'.

C. Performance of Gradient-Leaks

We first evaluate the performance of Gradient-Leaks, and the experiment results are shown in Table II. It should be noted that we fine-tune the parameters of the comparison methods to ensure that our reproduced results closely match those reported in their original paper. Note that we do not report similar results using Purchase dataset in this section due to space limits, but present some results in Section V-H to show the impact of the number of classes.

For the three native target models, we can observe that Gradient-Leaks performs better than Shadow Attack, ML-Leaks, Output Attack, and M-Entropy Attack, achieving the improvements of the attack precision by 10.7%, 11.3%, 7.9%, and 5.7%, respectively. To be more specific, when attacking the LR models, our attack achieves a mean attack precision of 0.608, while the attack performances of the comparisons are just around the random guess, which is higher than that of the comparisons by 13.5%. The mean recall of our attack is 0.615. When it comes to the RF models, Gradient-Leaks does not always get the best attack performance. For the RF model trained on MNIST dataset, M-Entropy Attack achieves the best attack precision of 0.647, which is higher than ours by 10.9%. When facing the RF models trained on other datasets, our attack gets precision improvements compared with the other three comparisons by 4.3%, 5.6%, and 12.8% respectively. As for the recall metric, M-Entropy Attack gets the best result of 0.808, which is higher than ours by 16.7%. When attacking against the DNN models, Gradient-Leaks gets the best attack performance. For the DNN models of binary classification tasks, although all the existing attacks perform similarly to random guess whose precisions are slightly higher than 0.5, we can obtain the attack precisions of 0.548 and 0.576 for Adult and Bank datasets, respectively.

As for the target models trained by MLaaS platforms, our attack still performs better than the comparison methods, except for the target models trained on MNIST dataset. When attacking the binary classification models, Gradient-Leaks performs much better than the comparison methods. Especially, for the model trained by BigML on Adult dataset, our attack can distinguish the members from non-members with a precision of 0.561 and a recall of 0.718. In this case, the attack precision of our model is higher than the comparison methods by 10.2%, 10.1%, 6.1%, and 3.2%, respectively.

TABLE II
ATTACK PERFORMANCE COMPARISONS

Target Model	**LR**			**RF**			**DNN**			**BigML**			**Amazon ML**		
Dataset	Adult	Bank	MNIST	Adult	Bank	MNIST	Adult	Bank	MNIST	Adult	Bank	MNIST	Adult	Bank	MNIST
Metric	**Attack Precision**														
Shadow Attack	0.514	0.509	0.564	0.562	0.574	0.563	0.505	0.512	0.557	0.509	0.508	0.597	0.536	0.547	0.524
ML-Leaks	0.512	0.502	0.533	0.536	0.552	0.604	0.504	0.506	0.572	0.507	0.519	0.612	0.523	0.533	0.541
Output Attack	0.505	0.517	0.564	0.529	0.535	0.521	0.523	0.516	0.568	0.529	0.538	0.637	0.523	0.548	**0.714**
M-Entropy Attack	0.560	0.505	0.535	0.538	0.589	**0.647**	0.532	0.541	0.607	0.542	0.527	**0.651**	0.564	0.568	0.668
Gradient-Leaks	**0.612**	**0.566**	**0.648**	**0.596**	**0.617**	0.576	**0.548**	**0.576**	**0.613**	**0.561**	**0.583**	0.629	**0.606**	**0.609**	0.583
Metric	**Attack Recall**														
Shadow Attack	0.427	0.528	0.517	0.741	0.628	0.673	0.504	0.579	0.506	0.622	0.621	0.691	0.573	0.562	0.534
ML-Leaks	0.524	0.462	0.557	0.704	0.660	0.615	0.554	0.596	0.546	0.706	0.552	0.619	0.525	0.518	0.579
Output Attack	0.516	0.594	0.635	0.778	0.741	0.683	0.702	0.774	0.674	0.667	0.713	0.501	0.537	0.512	0.576
M-Entropy Attack	**0.724**	**0.761**	**0.822**	**0.828**	**0.813**	**0.785**	**0.816**	**0.880**	**0.764**	**0.827**	**0.845**	**0.853**	**0.730**	**0.781**	**0.804**
Gradient-Leaks	0.625	0.578	0.614	0.715	0.697	0.608	0.627	0.701	0.723	0.718	0.674	0.637	0.572	0.631	0.565

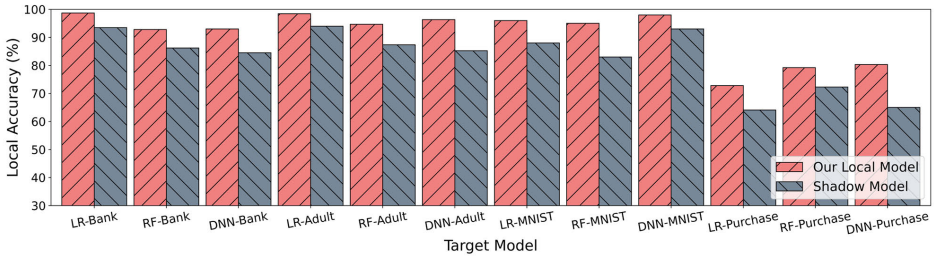


Fig. 4. Prediction accuracy comparison between our local model and the shadow model.

However, Gradient-Leaks is not always better than Output Attack and M-Entropy Attack, especially for the multi-class models trained on MNIST dataset, where the attack precision of our attack is lower than that of these two MIAs. For the target models trained by Amazon ML, our attack precision and recall are lower than that of Output Attack by 13.8% and 1.9%. Even so, Gradient-Leaks still performs better than Shadow Attack and ML-Leaks, reaching the improvements on the attack precision of 11.3% and 7.8%, respectively. As for the recall metric in this case, our attack exhibits a significantly lower recall compared to the M-Entropy Attack. It also shows a slightly lower recall than ML-Leaks and Output Attack, with a difference of approximately 2.5%. However, our attack's recall remains higher than Shadow Attack by 5.8%.

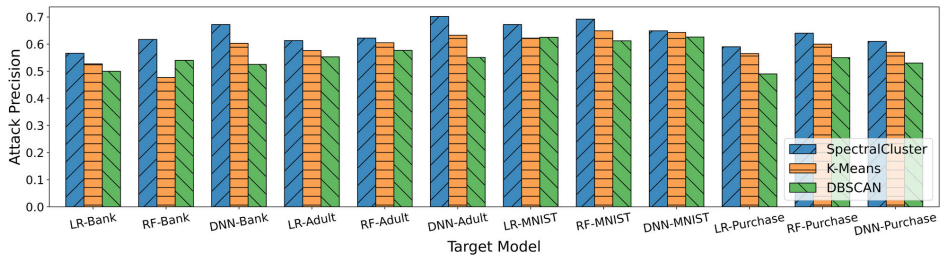
From the results, we can see that Gradient-Leaks outperforms in general the 4 comparisons against most target models. One possible reason is that our local model is better than the shadow model based methods at approximating the target model's prediction behavior around the target record (the results shown in Fig. 4 can also demonstrate this point of view). Therefore, MIAs with our local model can achieve a more precise performance. Moreover, Gradient-Leaks leverages the approximate gradient of the target record to perform MIAs. Compared with the prediction probability used by existing MIAs, the gradient can reflect the target model's prediction behavior to the target record from a more fine-grained perspective.

TABLE III
PREDICTION DIFFERENCE ON TARGET DATA RECORD

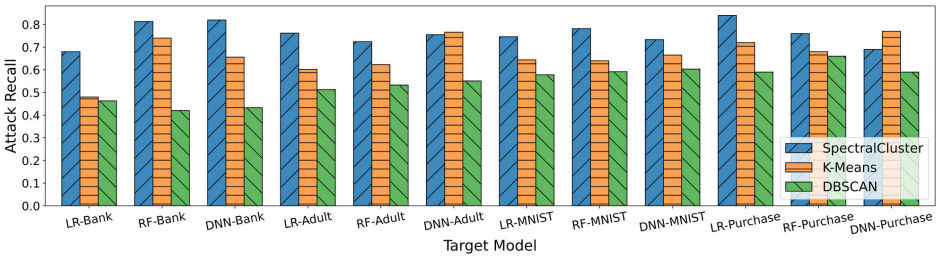
Target Model		**Prediction Difference (L1 Norm)**	
Dataset	Model	Shadow Model	Our Local Model
Adult	LR	3.86×10^{-2}	1.53×10^{-2}
Adult	RF	2.47×10^{-1}	1.15×10^{-1}
Adult	DNN	1.79×10^{-1}	1.76×10^{-2}
Bank	LR	2.96×10^{-2}	1.67×10^{-3}
Bank	RF	3.83×10^{-1}	4.32×10^{-2}
Bank	DNN	2.25×10^{-1}	1.95×10^{-2}
MNIST	LR	2.29×10^{-1}	8.95×10^{-2}
MNIST	RF	1.35×10^{-1}	4.06×10^{-3}
MNIST	DNN	7.14×10^{-2}	9.89×10^{-5}
Purcahse	LR	3.32×10^{-1}	4.12×10^{-2}
Purcahse	RF	6.42×10^{-2}	5.87×10^{-6}
Purcahse	DNN	1.34×10^{-1}	3.55×10^{-4}

D. Evaluation of Local Fidelity of Local Models

In this section, we evaluate the local fidelity of our local models on 12 different target ML models. We use the prediction difference between our model and the target model to measure the local fidelity of our local models. By comparing the prediction difference between our local models and the shadow models with the target model respectively, we could



(a) Attack Precision



(b) Attack Recall

Fig. 5. The impact of the algorithm of the attack models on inference attacks.

evaluate which model could better mimic the prediction behavior of the target model around the target sample. For each target model, we randomly select 100 samples from its training set as the target records. Then we derive the mean prediction difference and the mean local accuracy of the target records with respect to the shadow model and our local model. The prediction difference of the target records between the shadow model and our local model is shown in Table III.

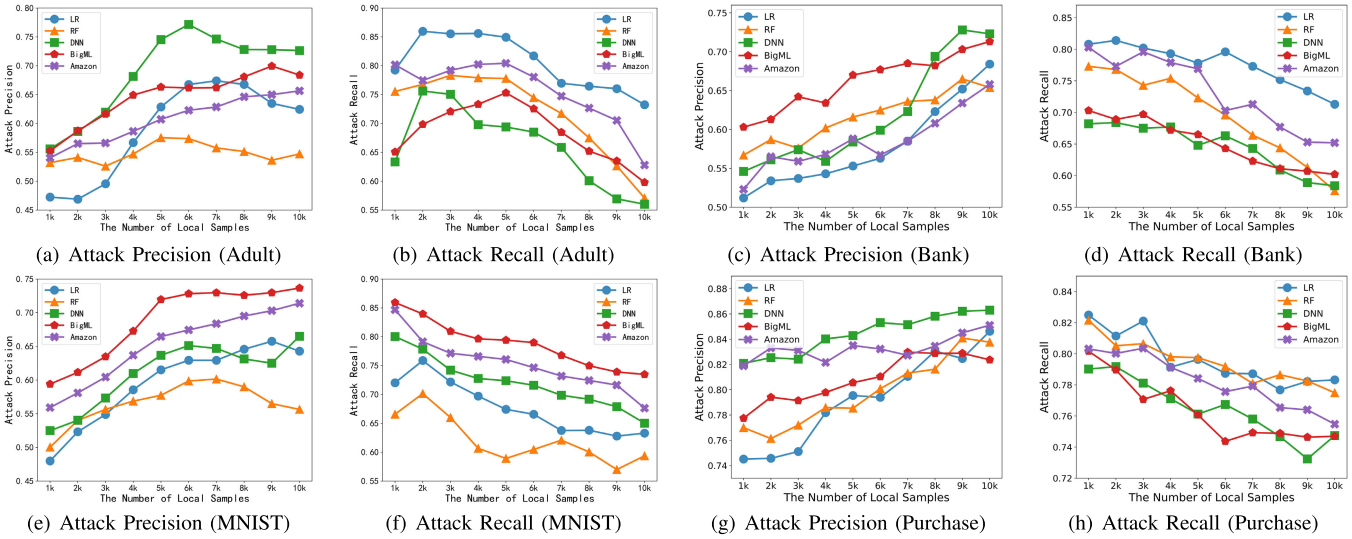
For the LR and RF models trained on Adult dataset, the prediction difference of the local model is around half that of the shadow model. However, for the DNN model, the prediction difference of our local model can be $10^{-1}\times$ smaller than that of the shadow model. Regarding the Bank dataset, the prediction difference of our local model is approximately half that of the shadow model with respect to the LR model, similar to the findings on the Adult dataset. However, for RF and DNN models, our local model demonstrates a better performance, exhibiting an improvement of up to $10\times$ compared to the shadow models. As for the MNIST dataset, the ratio of the prediction difference between the local model and the target model can reach $10^{-2}\times$, when Gradient-Leaks attacks the target models trained by LR and RF algorithms. When attacking against the DNN model, the ratio can be further reduced to $10^{-3}\times$. When considering the Purchase dataset, our method displays a significant difference in prediction difference compared to the shadow model across all target models, spanning multiple orders of magnitude. Notably, in the case of the RF model, the prediction difference achieved by our local model can be as much as $10^{-4}\times$ smaller than that of the shadow model.

From the experiment results we can find that when treating with DNN models, our local model always approximates the target model much better than the shadow model. The main reason is that the training process of DNN models involves randomness, e.g., on the initial parameters and the gradient orientation. Even we train the two DNN models with the same structure, training set, and the hyper-parameters, these two models would not be exactly the same as each other. Furthermore, since the training set of the shadow model is different from that of the target model, it will make the difference between the shadow model and the target model even larger. Consequently, the prediction of the shadow model will be far away from the target model.

Fig. 4 shows the local accuracy of the shadow model and the local model on the same set of data records. Over all target models, our local models achieve a mean local accuracy of 91.3%, which is higher than that of the shadow models by 10.1%. Especially, for DNN target models, our models perform better than the shadow models by 11.5%, 10.1%, 6.2% and 23.5% on the Adult, Bank, MNIST and Purchase datasets, respectively. From the results, we can find that our local models are more precise than the shadow model all time. This is because that the shadow model attempts to imitate the target model from a global perspective, while our local model aims to locally approximate the target model around a given record. It is easy to learn the prediction behavior of a model within a limited range of the data space.

E. Impact of Attack Model's Algorithm

To evaluate the impact of the algorithm of attack models, we test the attack performance with three attack models trained by different unsupervised clustering algorithms, including Spectral Cluster, K-Means, and DBSCAN. Based on the experiment results illustrated in Fig. 5, it is evident that the attack model developed using the Spectral Cluster approach exhibits the highest level of effectiveness against all target models. It achieves a mean precision of 0.683 and a mean recall of 0.781. On the other hand, the attack model trained with DBSCAN performs the worst, with its precision and recall being 10.2% and 24.7% lower than the Spectral Cluster model, respectively. The overall precision of the DBSCAN model barely exceeds random guessing, measuring only 0.536. Regarding the attack model trained using the K-Means algorithm, it demonstrates a mean attack precision of 0.587 and a mean recall of 0.659.



(a) Attack Precision (Adult) (b) Attack Recall (Adult) (c) Attack Precision (Bank) (d) Attack Recall (Bank)

(e) Attack Precision (MNIST) (f) Attack Recall (MNIST) (g) Attack Precision (Purchase) (h) Attack Recall (Purchase)

Fig. 6. The impact of the number of local samples.

When attacking DNN target models, the precision metric displays the most significant performance variability among different unsupervised clustering algorithms used to construct the attack models as shown in Fig. 5(a). For instance, in the case of inferring against the DNN model trained on Bank dataset, the attack precision of Spectral Cluster model is higher than that of K-Means model and DBSCAN model by 6.9% and 14.7%, respectively. However, all attack models behave relatively similarly when attacking against LR target models. For instance, the attack precision of Spectral Cluster model is higher than that of K-Means model and DBSCAN model by 3.9% and 6.6%, respectively, when attacking the LR model trained on Bank dataset.

The experiments show that the membership features extracted from the approximate gradient contain the information about the membership property of the given data, which can be used to determine whether the given data is in the target model's training data or not. Therefore, no matter which ML algorithm the target model employs, Gradient-Leaks can breach the membership privacy of the target model with a suitable attack model.

F. Impact of Number of Local Samples

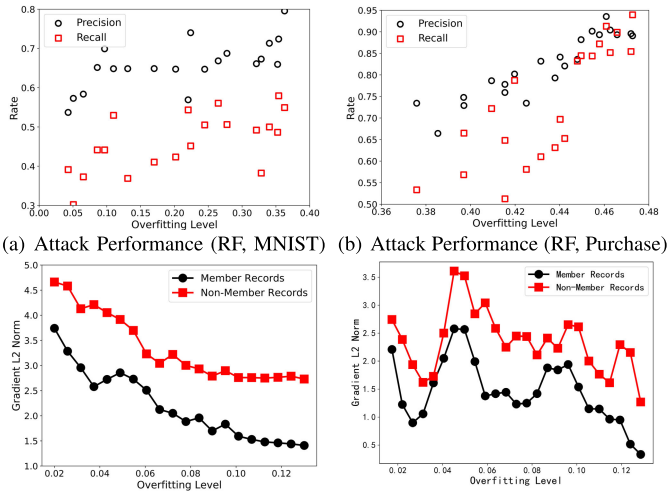
To quantify the impact that the number of local samples has on the performance of Gradient-Leaks, we perform 5 trials of the inference attacks on five different types of ML models with 1,000∼10,000 samples. In order to make it easier to observe the trend of the attack performance with the varying number of local samples, we average and then smooth the experimental results.

Fig. 6 shows the relationship between the number of local samples and the attack performance of Gradient-Leaks. In general, as the number of local samples increases, the precision of our inference attack becomes more accurate, which is not the case for the recall metric. For all the target models trained on Adult dataset, the mean attack precision is 0.529 and the mean recall is 0.739 when the number of local samples is 1,000. When we set the number of local samples to 10,000, the mean attack precision increases by 11.9% while the mean recall decreases by 10.7%. As shown in Fig. 6(a) and 6(b), when the number of local samples exceeds 5,000, the increasing samples cannot lead to a significant improvement of the attack precision, even when the attack recall continues decreasing. Regarding the Bank dataset, the performance of the attack with varying numbers of local samples is illustrated in Figs. 6(c) and 6(d). Analyzing the results, we can see that as the number of local samples increases from 1,000 to 10,000, the attack precision of Gradient-Leaks improves from 0.550 to 0.686, while the attack recall decreases from 0.763 to 0.634. However, after surpassing 7,000 local samples, the increasing trend in attack precision gradually diminishes, while the recall continues to decline fast.

The main reason is that the Adult and Bank datasets are quite simple and thus the prediction behavior of the model trained on these dataset is nontrivial for our local model to approximate. A small amount of samples will be sufficient for the local models to mimic the target model well. Nevertheless, with a large amount of samples, the local model will be overfitted around the given record and lose its generalization. In this case, the approximate gradient of the given record is in the way to performance degradation of Gradient-Leaks.

As shown in Figs. 6(e) and 6(f), when attacking the target models trained on MNIST dataset, Gradient-Leaks can achieve a mean attack precision and recall of 0.534 and 0.778, respectively, with only 1,000 local samples. As the number of local samples increases, the mean precision increases by 29.8%, while the mean recall decreases by 12.1%. As depicted in Figs. 6(g) and 6(h), when our MIA is applied to target models trained on the Purchase dataset, remarkable results are observed. Even with only 1,000 local samples, our MIA can achieve a mean attack precision of 0.778 and a mean recall of 0.803. Nevertheless, as the number of local samples increases to 10,000, the mean precision exhibits an improvement of 5.73%, while the mean recall exhibits a decrease of 6.24%.

The MNIST and Purchase datasets present a higher level of classification complexity compared to the Adult and Bank datasets. The behavior exhibited by the target models trained on these datasets is more intricate, making it challenging for



(a) Attack Performance (RF, MNIST) (b) Attack Performance (RF, Purchase)

(c) Gradient $L2$ Norm (DNN, MNIST) (d) Gradient $L2$ Norm (DNN, Purchase)

Fig. 7. The impact of overfitting.

our local model to accurately mimic their behavior. Consequently, to obtain a more precise local model, a large number of local samples is required to interact with the target model and gather additional insights into its prediction behavior. As our local model better approximates the target model, the gradient approximation becomes increasingly similar to that of the target model. This leads to an enhancement in the attack precision of our Gradient-Leaks. The results obtained from the Adult and Bank datasets lend support to our perspective.

G. Impact of Overfitting

In this section, we evaluate the impact of overfitting of target models on the performance of Gradient-Leaks. To achieve this, we perform our inference attack against a series of target models trained with different parameters, while the model's training data and training algorithm are kept unchanged. In order to quantify the overfitting level of the target model, we use the difference between its prediction accuracy on the training set and testing set as an indicator. The results in Figs. 7(a) and 7(b) demonstrate the relationship between the attack performance of Gradient-Leaks and the overfitting level of the target models constructed by RF algorithm on the MNIST dataset. It is obvious that with the overfitting level increasing, the target models are more vulnerable to MIAs. For instance, when the target model has an overfitting level of 4.38%, our attack achieves a relatively low precision and recall of 0.537 and 0.301, respectively. However, Gradient-Leaks achieves the precision and recall both around 0.9 when the overfitting level exceeds 45%.

In addition, we carry out further experiments to explore the relationship between the overfitting level of the target model and the norm of our approximate gradient. We train two sets of DNN target models on Purchase and MNIST respectively, and then get different overfitting levels by adjusting the number of each model's training epoch. We further leverage Gradient-Leaks to derive the approximate gradients for one set of training records (i.e. member records) and another set of testing records (i.e. non-member records), respectively. Then

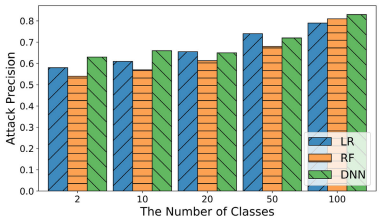


Fig. 8. The impacts of the number of classes (Purchase).

we average the approximate gradients respectively. As shown in Figs. 7(c) and 7(d), the approximate gradient norms of both member and non-member records decrease with an increasing overfitting level, but the norm of the members declines faster than that of the non-members. When the overfitting level exceeds a certain degree, the gradient norm of non-members will be gradually stable, which is not the case for member instances. Therefore, the difference of the approximate gradient norm between members and non-members will increase as the model overfitted more severely. Besides, it is obvious that member records have smaller gradient norm values than non-member records do.

Our experiments illustrate that overfitting can increase the risk of membership privacy to an ML model's training data. With the overfitting level of a model increasing, the difference of our approximate gradient between member and non-member records becomes widened. As such, Gradient-Leaks can separate the member records from non-members and achieves a relatively high attack performance against an overfitted model.

H. Impact of Number of Classes

The number of output classes of the target model contributes to how much the ML model leaks. With more output classes, Gradient-Leaks can obtain more information about the prediction behavior of the target model and thus can derive more detailed local approximate gradient.

To evaluate the impact of the number of output classes, we train a series of target models using LR, RF, and DNN on the Purchase dataset with 2, 10, 20, 50, 100 classes. Fig. 8 shows the attack precision against different target models. For each type of ML models, the performance of Gradient-Leaks has a significant improvement as the number of output classes increases. Specifically, the attack precision of our attack is 0.582 when the LR model has 2 output classes, while increasing to 0.784 when the class number is 100. As for the target models trained by RF algorithm, the performance of our attack has the most significant improvement and the attack precision increases by 0.263. For DNN target models, our attack precision increases from 0.638 to 0.823.

From the results we can observe that, models with fewer output classes leak fewer information about their membership property in the training data. As the number of classes increases, the target models need to learn more distinctive features from the training data to achieve a higher classification accuracy and to remember more about their training data. As a consequence, ML models with more outputs may leak more information and suffer from more severe MIAs.

VI. Conclusion

In this paper, we have presented Gradient-Leaks, the first membership inference attack (MIA) against ML models with mere black-box access. Gradient-Leaks leverages a linear ML model trained around the target record to derive the approximate gradient with respect to the target model and further extracts the vital membership features to facilitate the membership inference. Extensive experiments on different types of ML models with real-world datasets show that Gradient-Leaks can achieve better performance compared to the state-of-the-art MIAs, even without any prior knowledge about the target model and its training data. We believe our work may deepen the understanding of the training data privacy risks of ML models in practical settings and shed light on exploiting more effective countermeasures against MIAs.

References

[1] F. Li et al., "Mask DINO: Towards a unified transformer-based framework for object detection and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 3041–3050.

[2] W. X. Zhao et al., "A survey of large language models," 2023, *arXiv:2303.18223*.

[3] Y. Yang, C. Huang, L. Xia, and C. Li, "Knowledge graph contrastive learning for recommendation," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 1434–1443.

[4] N. Aafaq, N. Akhtar, W. Liu, M. Shah, and A. Mian, "Language model agnostic gray-box adversarial attack on image captioning," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 626–638, 2023.

[5] Y. Shen, X. He, Y. Han, and Y. Zhang, "Model stealing attacks against inductive graph neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 1175–1192.

[6] T. Zhu, D. Ye, S. Zhou, B. Liu, and W. Zhou, "Label-only model inversion attacks: Attack with the least information," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 991–1005, 2023.

[7] L. T. Phong and T. T. Phuong, "Privacy-preserving deep learning via weight transmission," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 11, pp. 3003–3015, Nov. 2019.

[8] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[9] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," *ACM Comput. Surv. (CSUR)*, vol. 54, no. 11s, pp. 1–37, 2022.

[10] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jegou, "White-box vs black-box: Bayes optimal strategies for membership inference," in *Proc. ICML*, 2019, pp. 5558–5567.

[11] B. Wu et al., "Characterizing membership privacy in stochastic gradient Langevin dynamics," 2019, *arXiv:1910.02249*.

[12] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 739–753.

[13] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Found. Symp. (CSF)*, Jul. 2018, pp. 268–282.

[14] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "ML-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.

[15] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in classification models," 2020, *arXiv:2002.12062*.

[16] G. Liu, C. Wang, K. Peng, H. Huang, Y. Li, and W. Cheng, "SocInf: Membership inference attacks on social media health data with machine learning," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 5, pp. 907–921, Oct. 2019.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1135–1144.

[18] J. Lee et al., "Wide neural networks of any depth evolve as linear models under gradient descent," in *Proc. NeurIPS*, 2019, pp. 8570–8581.

[19] R. Shokri, M. Strobel, and Y. Zick, "Exploiting transparency measures for membership inference: A cautionary tale," in *Proc. AAAI Workshop Privacy-Preserving Artif. Intell.*, vol. 13, 2020, pp. 1–11.

[20] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *Proc. USENIX Secur.*, 2021, pp. 2615–2632.

[21] Y. Liu, Z. Zhao, M. Backes, and Y. Zhang, "Membership inference attacks by exploiting loss trajectory," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 2085–2098.

[22] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *Proc. ICML*, 2021, pp. 1964–1974.

[23] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "LOGAN: Membership inference attacks against generative models," in *Proc. Privacy Enhancing Technol.*, 2019, pp. 133–152.

[24] K. S. Liu, C. Xiao, B. Li, and J. Gao, "Performing co-membership attacks against deep generative models," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2019, pp. 459–467.

[25] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 691–706.

[26] S. Truex, L. Liu, M. E. Gursoy, W. Wei, and L. Yu, "Effects of differential privacy and data skewness on membership inference vulnerability," in *Proc. 1st IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl. (TPS-ISA)*, Dec. 2019, pp. 82–91.

[27] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr, "Membership inference attacks from first principles," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 1897–1914.

[28] H. Yan et al., "Membership inference attacks against deep learning models via logits distribution," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 5, pp. 3799–3808, Sep. 2023, doi: 10.1109/TDSC.2022.3222880.

[29] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 880–895.

[30] C. Molnar. (2019). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. [Online]. Available: https://christophm.github.io/interpretable-ml-book/

[31] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J.*, vol. 37, no. 2, pp. 233–243, Feb. 1991.

[32] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Jan. 2010.

[33] R. Boney et al., "Regularizing trajectory optimization with denoising autoencoders," in *Proc. NeurIPS*, 2019, pp. 2855–2865.

[34] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with VQ-VAE-2," in *Proc. NeurIPS*, 2019, pp. 14837–14847.

[35] H.-S. Lee, Y.-D. Lu, C.-C. Hsu, Y. Tsao, H.-M. Wang, and S.-K. Jeng, "Discriminative autoencoders for speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 5375–5379.

[36] J.-C. Chou, C.-C. Yeh, H.-Y. Lee, and L.-S. Lee, "Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations," in *Proc. Interspeech*, Sep. 2018, pp. 501–505.

[37] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Trans. Syst. Man, Cybern., B, Cybern.*, vol. 29, no. 3, pp. 433–439, Jun. 1999.

[38] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. ACM SIGKDD*, 1996, pp. 226–231.

[39] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[40] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2012.

[41] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019, pp. 8024–8035.