ML Defense: Against Prediction API Threats in Cloud-Based Machine Learning Service

Jiahui Hou
Univ. of Sci. and Tech. of China
Illinois Institute of Technology
jhou11@hawk.iit.edu

Jianwei Qian
Illinois Institute of Technology
jqian15@hawk.iit.edu

Yu Wang
U. of North Carolina at Charlotte
Yu.Wang@uncc.edu

Xiang-Yang Li
University of Science and Technology of China
xiangyangli@ustc.edu.cn

Haohua Du
Illinois Institute of Technology
hdu4@hawk.iit.edu

Linlin Chen
Illinois Institute of Technology
lchen96@hawk.iit.edu

ABSTRACT

Machine learning (ML) has shown its impressive performance in the modern world, and many corporations leverage the technique of machine learning to improve their service quality, *e.g.,* Facebook's DeepFace. Machine learning models with a collection of private data being processed by a training algorithm are deemed to be increasingly confidential. Confidential models are typically trained in a centralized cloud server but publicly accessible. ML-as-a-service (MLaaS) system is one of running examples, where users are allowed to access trained models and are charged on a pay-per-query basis.

Unfortunately, recent researchers have shown the tension between public access and confidential models, where adversarial access to a model is abused to duplicate the functionality of the model or even learn sensitive information about individuals (known to be in the training dataset). We conclude these attacks as *prediction API threats* for simplicity.

In this work, we propose **ML defense**, a framework to defend against prediction API threats, which works as an add-on to existing MLaaS systems. To the best of our knowledge, this is the first work to propose a technical countermeasure to attacks trumped by excessive query accesses. Our methodology neither modifies any classifier nor degrades the model functionality (*e.g.,* rounds results). The framework consists of one or more simulators and one auditor. The simulator learns the hidden knowledge of adversaries. The auditor then detects whether there exists a privacy breach. We discuss the intrinsic difficulties and empirically state the efficiency and feasibility of our mechanisms in different models and datasets.

CCS CONCEPTS

• **Security and privacy → Security services**; • **Computing methodologies → Machine learning**.

KEYWORDS

Machine learning as a service; Privacy and security

ACM Reference Format:
Jiahui Hou, Jianwei Qian, Yu Wang, Xiang-Yang Li, Haohua Du, and Linlin Chen. 2019. ML Defense: Against Prediction API Threats in Cloud-Based Machine Learning Service. In *IEEE/ACM International Symposium on Quality of Service (IWQoS '19), June 24–25, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3326285.3329042

1 INTRODUCTION

Increasingly, machine learning (especially deep learning) has become a foundation of online cloud computing services, which provides predictions for many tasks, such as medical diagnosis [7], and facial recognition [27]. The significantly promising performance promotes many Internet company giants such as Google and Amazon to build machine learning as a service (MLaaS) systems. In such a system, users are allowed to access a trained model and charged on a pay-per-query basis.

ML classifiers (namely models) are always deemed to be confidential, especially those used in security-critical areas such as medical diagnosis. Specifically, individual health data, photos, their personal activities, *etc.*, on which models were trained, make the ML classifiers sensitive and private. For example, the training sets used in face recognition are pictures of individuals' faces.

In the context of privacy implications, existing solutions, *i.e.,* privacy-preserving training [1] and privacy-preserving prediction [4, 12], are far from enough. Recent researches demonstrated that adversaries can infer valuable information (either model relevant information or training data relevant information) through continuously querying a confidential ML model via a prediction query interface. In the work of Shokri *et al.* [25], they demonstrate that *membership inference attacks* are possible in machine learning: given a data record and black-box access (accessible only via a prediction API), one can determine whether the record is used in the training set (*i.e.,* members or non-members). Further still, Tramèr *et al.* [29] investigate *model extraction attacks* via continuously using prediction APIs such that the functionality of a trained model would be duplicated. For convenience, we refer to these attacks later in the paper as *prediction API threats.* Shokri *et al.* [25] exhibit that natural countermeasures (*e.g.,* round the prediction results) to membership inference attacks are ineffective and only experimentally evaluated in a single setting, and it remains unclear if they pose threats to other attacks. In the other context of model extraction attacks, Tramèr *et al.* [29] highlight the need for new model extraction countermeasures. Consequently, new countermeasures or solutions need to be further explored.

The goal of our work is to defend against these ML attacks which are initiated by excessive query accesses. Facing those unpredictable ML attacks, we need to overcome the following challenges. First, finding countermeasures to one existing attack is an open question and needs to be further explored. It is difficult to find a universal countermeasure (*i.e.,* not limited to one specific type of attacks) to fulfill the protection of arbitrary target models and training datasets without adversaries' background knowledge. Second, devising a robust and scalable countermeasure while guaranteeing the efficiency and quality of service (*i.e.,* sustaining the classifiers' predictive power) is a daunting challenge. Detecting attacks by brute force in running all existing attack algorithms is exceptionally inefficient.

Classifiers reveal information for two main reasons.

- **Memorability.** Machine learning indeed can extract insights from data. Specifically, the relation and correlation embedded in the training data are gathered in the model. Thus, classifiers inherently reveal some information. For example, records which are classified with high confidence are always assumed to have a statistically similar population property (*i.e.,* similar distribution) with the training data.
- **Accessibility.** More specific knowledge inferred by the outputs of queries makes the attacks possible and more effective than others (the one with fewer queries) even if both are implemented with the same attack algorithms. Shokri *et al.* [25] report that numbers of predicted probabilities output obtained through an accessible query interface can be used to identify whether a given record is used in training set. Demonstrated in [29], adversaries have a better performance in model extraction attack if increasing the number of query accesses to the model.

It is worth noting that we are not intended to change any trained model but rather interested in preventing useful information from being revealed when allowing access to the trained model. That is, we do not redesign any machine learning algorithm or mitigate the training set. Consider the following example: there is a facial recognition model (*e.g.,* FaceID) which has better performance in recognizing faces than others, even if all are trained on the same training set. The model is carefully designed and well deployed; thus, it does not make sense to reconstruct a model that possibly damages the predictive power. In our work, we focus on how to build an add-on to discover information leakage and protect valuable classifiers when allowing query access to the model. We can extrapolate and uncover specific hidden information implied in prediction output vectors. By doing this, the secret sauce (*i.e.,* both dataset and training algorithm), which makes the facial recognition model keep ahead of others, remains confidential. Therefore, the type of problem we focus on is related to the second reason (accessibility). To deal with it, we design **ML defense**, which consist of one or more *simulators* and one *auditor.* A simulator reconstructs the adversaries' information which is obtained via an accessible query interface but possibly contributes to ML attacks. To do this, we use a hidden representation learner to uncover useful properties of obtained data. Later, we use an *auditor* to audit how different the adversaries' data is from the target ones (*i.e.,* the training dataset). An auditor learns a function $au : \{\mathbf{M}_D, \mathbf{M}_A\} \rightarrow \{0, 1\}$, where $\mathbf{M}_D$ is a representation of training data and $\mathbb{M}_A$ is a representation learned by adversaries. This function works as a privacy measurement, and it is considered to be a privacy breach if the value is greater than a threshold (*i.e.,* adversaries almost learn the data distribution hidden in the training data). In that case, the auditor will begin to reject any query access.

In this paper, we investigate countermeasures to attacks developed by continuous accesses via a prediction query interface, *i.e.,* ML prediction API threats. Note that the proposed **ML defense** is independent of the existing classifiers. We first assume the attacker has no prior knowledge of the classifier (*i.e.,* its parameters) and the training datasets. In the case of black-box access, we first provide the insights on which our methodology is based and later we build our prototype by setting up a framework using a simulator and an auditor. We also evaluate our method on different ML classifiers such as logistic regression and deep neural networks.

Our work evinces that existing privacy issues faced by machine learning applications can be somewhat mitigated. The main contributions of our work are as follows. First, we study and analyze the fundamental reasons for successful prediction API threats. The insights shown in the following section describe the feasibility of our work. Second, we put forward a prototype framework against existing ML prediction API threats that, to the best of our knowledge, has not been theoretically studied before. Our methodology is shown as an add-on to existing MLaaS systems. Last, we further show the effectiveness and efficiency of our methodology: we successfully defend against ML attacks via prediction APIs. For instance, we prevent membership attacks which are implemented via neural networks and make the attack accuracy degrade to 52%. We believe our methodology can be applied to most existing commercial ML classifiers, defending against general attack strategies.

The rest of our paper is organized as follows: Section 2 introduces the necessary background and references. We define the system model and attack/defense problems in Section 3. Section 4 provides the insights from existing works and addresses the design space. In Section 5, we state our problem mathematically. Section 6 presents the protocol for ML prediction API threats. Section 7 shows the performance evaluations, followed by discussion in Section 8. Finally, Section 9 concludes with potential future works.

2 BACKGROUND AND RELATED WORKS

Machine learning plays an increasingly significant role in numerous areas such as medical management [8], financial industry [18]. Many corporations and organizations are building MLaaS systems. However, privacy and security concerns are one of the most important issues in these security-critical areas.

2.1 Existing Attacks

Recently, researchers demonstrate that both the classifiers and training data can be inferred. More specifically, Tramèr *et al.* [29] showed that it is possible to steal machine learning models when only given

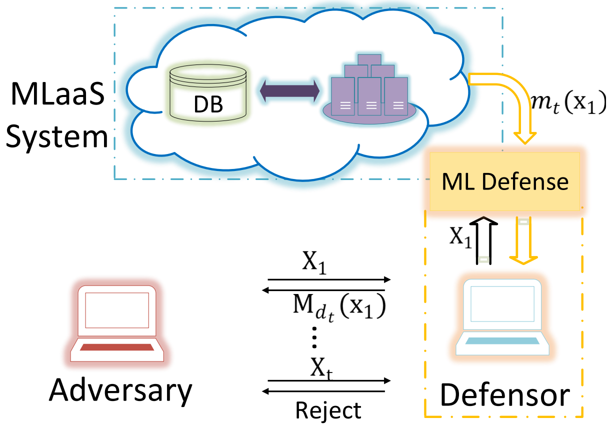


Figure 1: System Model Overview. ML defense works as an add-on for current MLaaS systems. To defend against prediction API threats, ML defense extends the outputs of a target classifier *i.e.,* from $m_t(x)$ to $M_{d_t}(x)$.

prediction results. The work of Ateniese *et al.* [2] declares that sensitive information or pattern of training data can be inferred by hacking into classifiers. In their work, they showed that it can infer whether an accent of speakers was originally employed in the voice recognition system. Later on, Fredrikson *et al.* [9] extended the work and devise the model inversion attack, which is able to reconstruct an image from a particular label in a facial recognition system. Shokri *et al.* [25] continued to show that it is possible to infer whether a certain record is in the training data by membership inference attacks. These attacks highlight the need for protecting classifiers and private data from leakage when using machine learning services. In this paper, we focus on the type of attacks whose success benefits from excessive query accesses such as member attacks and model extraction attacks.

2.2 Existing Defenses

In the query auditing problem [14, 21, 22], an auditor requires to decide whether to answer or deny the query when given a newly received query. There are some related works with privacy-preserving data access [15, 16]. Our work can be considered as an extension of the query auditing problem, where we support richer queries of different types, *i.e.,* machine learning, and deep learning. However, previous techniques cannot be applied since the query function is not any polynomial function and none of the state-of-the-art tools can be used to solve the solution space of the function without given an explicitly mathematical definition. A set of works are developed for privacy-preserving training and privacy-preserving classification. Homomorphic encryption is employed to protect training data and predictions in many works [4, 31]. For example, Bost *et al.* [4] show that a third party can compute encrypted predictions on encrypted medical data using fully homomorphic encryption. Zhang *et al.* [31] employ somewhat homomorphic encryption to protect private training data from leakages when an ML model is trained by a third party. Moreover, the work of Shokri and Shmatikov [24] introduces distributed collaborative learning to protect the privacy of the training data. In that system, multiple parties individually train their local models and collaboratively train a global model by sharing gradients of the local models through a parameter server. Collaborative learning is also extended in [6, 20, 30]. As shown in [9], differential privacy also plays an important role in privacy-preserving training. However, these protections are useless when we consider continuous query accesses. Specifically, no work has been done to deal with prediction API threats to the best of our knowledge. Those attacks are shown to be robust against some mitigation strategies (*e.g.,* using regularization, coarsening precision of the prediction vector) in [25, 29]. In this paper, we design and implement a practical defense mechanism against prediction API threats.

3 PRELIMINARIES

3.1 System Model

The system model is depicted in Fig.1. A machine learning as a service system is built on the cloud server. Specifically, there is a trained classifier m_t being stored in a MLaaS system. The trained classifier (*i.e.,* target model in ML attacks) m_t is kept confidential. As shown in Fig.1, our **ML defense** is an add-on to the MLaaS system. We do not alter any functionality of m_t but only extend the outputs. Any authorized users can input a query X, where X is a vector of prediction input. A *defensor*, who establishes an ML defense, will determine whether it is safe to answer the newly received query and then return $M_{d_t}(X)$.

3.2 Threat Model

The Defensor, who constitutes **ML defense** and works as a part of cloud-based machine learning service, is considered honest. They will audit and prevent any adversary from probing the trade secret hidden in the trained models. Users are always considered as semi-honest adversaries, who follow the protocol specifications but aim to infer sensitive information while being allowed to access a profitable model. Regarding the knowledge of adversaries, we only consider one scenario, *black-box access*, where adversaries have no prior knowledge of target models' parameters or training data. They can query a model only via prediction APIs. We assume there is no collusion among multiple users in our work. Otherwise, we will consider all users as one giant user, which means the Defensor will take the knowledge of all users as the adversaries' dataset and will not repeatedly answer any query.

3.3 Attack & Defense

We first define the following sets and functions.

(1) $\mathcal{D}_t$: the set of all training samples on which the target model is trained with respect to an ML task t.
(2) $\mathcal{S}$: the set of all samples in the sample space (*e.g.,* all digit images).
(3) C_t: the union set of classes for the task t where $|C_t|$ is defined as the number of the mutually exclusive classes.
(4) $\mathcal{Q}_t = \{X | X \in \mathcal{S}$ and $X \in \mathbb{R}^d$ is an input with respect to the ML task $t\}$. $\mathcal{Q}_t$ is the set of all queries proposed by adversaries and sampled from $\mathcal{S}$. Each ML classification task

assumes a data-generation distribution from which sample $\in \mathcal{D}_t$ is generated. Researchers believe that records which are classified with high confidence have a similar distribution as $\mathcal{D}_t$ [25]. Since we do not know the exact query space of adversaries, we approximate Q_t by the union of samples classified with high confidence.

(5) $\mathcal{Y}_t$: the set of probabilities output of a c-class classifier with regard to the task t. Specifically, each element of $\mathcal{Y}_t$ is a confidence value vector $y \in \mathbb{R}^c$ where $c = |C_t|$.

(6) $\mathcal{A}_t$: the set defined as the *adversaries' dataset*. If adversaries are only allowed to black-box access, the query set Q_t and its corresponding $\mathcal{Y}_t$ jointly establish $\mathcal{A}_t$.

Definition 1. *A model or a classifier for a classification task t is a function m_t that maps $\mathcal{S}$ to $\mathcal{Y}_t$.*

Definition 2. *Given an input $x \in \mathcal{S}$, the prediction API threats for a classification task t are defined as follows.*

- Model-related attacks are modeled as a function $g_{m_t}: x \times \mathcal{A}_t \rightarrow \hat{\mathcal{Y}}_t$.
- Training data-related attacks are defined as a function $g_{d_t}: x \times \mathcal{A}_t \rightarrow [0, 1]$.

Note that, $\mathcal{Y}_t$ is defined as the set of all prediction outputs of the target model m_t while $\hat{\mathcal{Y}}_t$ represents the outputs of adversaries' extracted model g_{m_t}. The ground-truth classifier of the corresponding training data set $\mathcal{D}_t$ is m_t, we then can use the total variation distance (denoted as f_d) to evaluate the performance of g_{m_t} and g_{d_t}, respectively. The output of f_d represents the probability of whether there exists a successful attack. In a general source data-related attacks, adversaries can determine whether it is one of the samples within the dataset or a mitigated one (which belongs to the same individual adult) with a given record.

Definition 3. *Given a training dataset $\mathcal{D}_t$ and a model m_t, an ML defense against prediction API threats is a function $M_{d_t}: \mathcal{S} \rightarrow \mathcal{Y}_t \cup \emptyset$, where $\emptyset$ represents the judgment that the current query is likely insecure. Specifically, let $\mathbb{P} = \max\{f_d(g_{m_t}(x), m_t(x)), g_{d_t}(x))\}$, if given an input x, depending on whether a prediction API threat proceeds,* ***ML defense*** *makes the following decisions:*

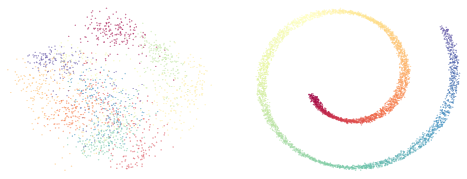
- x is *insecure*, *i.e.,* $\mathbb{P} \geq 1-\delta$, it returns $M_{d_t}(x) = \emptyset$. Here, $\mathbb{P}$ is a measurement for detecting any privacy leakage in the query access. δ is a privacy-related parameter, which is always of a large value in practice since a larger value represents better privacy.
- Otherwise, we have $M_{d_t}(x) = m_t(x)$ and x is denoted as a *secure query* in this case.

Our goal is to design such an ML defense, which is a countermeasure to the state-of-the-art ML attacks. Our **ML defense** prevents the adversaries from learning the hidden data-generating distribution of the training data. We will explain and address the design space in Section 4.

4 INSIGHT & DESIGN SPACE

4.1 Insights

Machine Learning Model is Based on the Training Data. Machine learning is about learning aspects of the unknown data-generating distribution from which the training data is sampled.



(a) Digits dataset (b) Swiss Roll dataset.

Figure 2: Manifolds of two different datasets. We respectively apply principal components projection (PCA[13]) to represent the digits and swiss roll dataset in two-dimensional space.

ML provides a predictive function that maps a feature vector, $\mathbf{X}$, to a set of response (*i.e.,* categories or real-value output vectors), $\mathbf{Y}$. The distribution of training data can be characterized by a probability density function, denoted $P(\mathbf{X},\mathbf{Y})$, which is determined by unknown parameters. Then, ML aims to best approximate the true parameters of the distribution. Indeed, superior training sets make a more effective classifier since more specific knowledge inferred by the training data is formed. Thus, it is fair to say that the machine learning model is exclusively dependent on training sets. In other words, we can keep the confidentiality of the model if adversaries cannot successfully learn the distribution of the dataset (*i.e.,* $P(\mathbf{X},\mathbf{Y})$).

Manifold Hypothesis in Machine Learning. Manifold is a topological space [5], which can be described by a collection of charts (*i.e.,* map), *e.g.,* $x^2 + y^2 = 1$ is a one-dimensional manifold (circle). Researchers speculate that the data-generating distribution (such as those involving images, video, text) is assumed to be highly concentrated in the region of embedded low-dimensional manifolds for many AI tasks [5, 23]. For example, a set of images associated with the same object class can form a low-dimensional manifold. From the perspective of manifolds, machine learning is used to capture aspects of manifolds and then disentangle the manifolds of different object classes. That is, data distribution can be represented by manifold. For example, Fig.2 depicts that data distribution is concentrated and represented in two-dimensional space. Besides, as shown in Fig.2, the differences embedded in training sets are finally presented in the manifolds, where we have different manifolds when given different data distribution.

Original Training Set v.s. Adversaries'. Given enough input records along with real-value prediction outputs, the work of [29] shows that using the similar optimization method used in the training phase (*i.e.,* to make an appropriate loss function converge to a global minimum) can achieve model extraction attack in a multiclass logistic regression setting. Moreover, a number of input records which are with high classification probability contribute to inferring whether a given record is a component of the training set (*i.e.,* members) [25]. In that work, it is assumed that records classified with high confidence have a similar distribution to the training records. That is, the data-generating distributions, from which the real training records and adversarial data are sampled, are similar. These insights raise a question: whether the similar distributions have an impact on successful attacks. As demonstrated in [25], they have a successful membership attack (up to 97%) on a standard

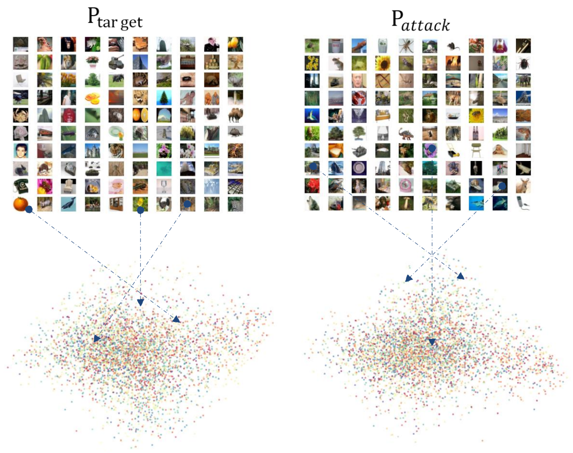


Figure 3: Comparison of manifolds. We exhibit the data distribution of the target training data and the data used for training a membership attack model in CIFAR-100. The top shows the 2D projection using a random 10×10 unitary matrix. The bottom is the two-dimensional manifold representation using PCA.

convolutional neural network which is trained on the CIFAR-100 dataset. We then use manifolds to visualize the data distribution. Fig.3 compares the distribution of the training data and the synthetic one on which the successful membership inference attack is trained. Indicated in Fig.3, similar data distributions likely have an impact on successful model extraction attacks. These results also empirically state that records classified with high confidence have a similar distribution with the real training set. Thus, it is reasonable to define adversaries' dataset to be the union of samples classified with high confidence, and compare its distribution with the distribution of the training dataset.

4.2 Design Space & Principles

Machine learning model is highly dependent on data-generating distribution from which the training records are sampled. As depicted in Fig.3, we extrapolate that similar data distributions likely have an impact on successful ML prediction API threats. We then leverage these opportunities and propose an ML defense protocol which audits each received query by taking training data and adversarial data (learned from ML query access) as inputs. The fundamental idea behind our defense is to prevent adversaries from recovering the similar data-generating distribution when facing a batch of responses to ML queries. The most challenging part is to compare those unknown data-generating distributions while ensuring efficiency and prediction accuracy. A careful protocol design is required to fulfill this purpose. Indeed, a successful ML defense system should respect some necessary principles:

- *Feasibility*: Our protocol should manage to defense against existing ML attacks which are triggered by excessive accesses. Also, our strategy should not damage the predictive power of any existing classifier.
- *Efficiency* and *Scalability*: Response time is important when providing ML services. Moreover, we need to make our protocol scalable such that it can be adaptive to different MLaaS systems.
- *Universality*: Facing a variety of profitable classifiers, our protocol should provide a universal and general countermeasure against ML models used on MLaaS systems. We need to design a strategy which is independent of existing ML models.

5 PROBLEM STATEMENT

As mentioned before, there can be privacy loss if the data distribution of training dataset is disclosed. Besides, the knowledge of data distribution contributes to any attack which leverages the technique of generative adversarial network (GAN) since the goal of GANs is to generate similar-looking samples to those in the training set (*i.e.,* ideally with the same distribution). Thus, we compare data distributions of adversaries' dataset (represented as $\mathcal{M}_{A_t}$) and the training set (represented as $\mathcal{M}_{D_t}$) in our work. Our problem is defined as follows.

Given a training dataset $\mathcal{D}_t$, a model m_t and an ML query x_l , to against prediction API threats, **ML defense** makes a decision based on $\mathbb{P}\left(\mathcal{M}^l_{A_t} = \mathcal{M}_{D_t}\right)$, where $\mathcal{M}^l_{A_t}$ is the representation of adversaries' dataset, *i.e.,* $\mathcal{A}^l_t = \mathcal{A}^{l-1}_t \cup x_l, y_l$. y_l is the corresponding prediction vectors of x_l. If the result of the measurement for detecting any privacy leakage in the query access is greater than a budget, which is controlled by a privacy-related parameter (*i.e.,* $\mathbb{P} \geq 1 - \delta$), we deny the query *i.e.,* $M_{d_t}(x) = \emptyset$. Otherwise, we return the prediction results, denoted as $M_{d_t}(x) = m_t(x)$. That is, we define privacy as $\mathbb{P}$, and it is considered a breach of privacy when two data distributions are similar.

6 ML DEFENSE PROTOCOL

6.1 Protocol Overview

To defend against prediction API threats, we propose a framework **ML defense** as depicted in Fig.4. In Section 4, we empirically provide evidence to show an ML classifier is mainly based on the data distribution. Motivated by these observations, our ML defense (*i.e.,* the Defensor) consists of two components: (1) a *simulator* that simulates and precisely represents unknown data distributions; (2) a *auditor* who strives to audit whether a received query is safe to answer by comparing two given data distributions which are represented by the simulators. Fig. 4 illustrates the workflow of our framework. Notably, the framework only works as an add-on when deploying a target model, and the output of the auditor is fed to the prediction outputs of the target model.

6.2 Simulator

The simulator is a function $s: X \to \mathcal{M}$ that characterizes a given dataset. One naïve approach is that we first define a data distribution P, and then learn the probability density of the data. This is often done by optimizing the parameters and maximize the log-likelihood, *i.e.,* $\max_\theta \frac{1}{m} \sum_{i=1}^{n} log P_\theta(X, Y)$, where $\mathbf{X}$ is input vector while $\mathbf{Y}$ is a set of response. The latent data distribution of adversaries' dataset $\mathcal{A}_t$ is then be captured by approximating a probability distribution

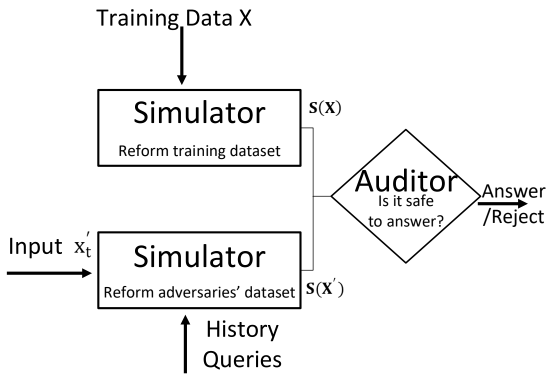


Figure 4: The workflow of ML defense. Our ML defense includes one or more simulators and one auditor. A simulator generates a good representation to simulate unknown data distribution hidden in a given dataset, followed by an auditor who audits privacy leakage by comparing two data distributions characterized by the simulators.

on $\mathcal{A}_t$. Many well-established methods such as kernel density estimation or Parzen window estimation are often used to estimate the density function of data distribution [3]. However, these methods derogate one of the essential design principles – efficiency due to the training time consumption. Therefore, it unlikely trains a machine learning model based on the dataset of adversaries. On the other hand, if the simulator is implemented with the idea of log-likelihood, it requires a very large number of examples to be close to a true log-likelihood [28].

When comparing two data distributions, we use the simulator to represent data distribution in low-dimensional space instead of comparing adversaries' dataset and the training dataset in high-dimension directly. This is because we suffer from *the curse of dimensionality* [10] in high-dimension where the distribution of the training dataset becomes more and more sparse. The ideal comparison should be applied in a low-dimensional feature space.

The well-known manifold hypothesis [23] puts forth that in many cases, such as those involving natural data, it forms a low dimensional manifold $\mathcal{M}$ in an embedding space. Particularly, empirical evidence is shown in Section 4 emphasizes that manifolds can capture the distributions of data samples. From this perspective, it is very likely there exist a manifold formed by adversaries' dataset which represents its data distribution in low-dimension. As an example of this approach, there are two distinct manifolds embedded in Euclidean space: one formed by adversaries' dataset $\mathcal{A}_t$, another formed by the training dataset $\mathcal{D}_t$. We apply the techniques of UMAP [19] in producing a low dimensional representation to approximate the manifold. This is because the technique arguably preserves more local structures with superior running time in comparison to previous techniques. The hidden data distribution can be kept with the technique of UMAP. We will show our simulator has excellent performance in preserving local structures when compared with PCA.

6.2.1 *Approximating the manifold.*

The approach is to use some simpler spaces in such a way it can resemble the topological properties of the dataset. These simplicial complexes are built from points, lines, triangles and other higher dimensional simplices as depicted in Fig.5. If data is uniformly distributed along the manifold, a neighborhood-graph can be simply built and laid out in lower dimension because the manifold can be well covered by balls with a suitable radius. Otherwise, we can find a metric if we assume the manifold has a Riemannian metric. Formally, let the input data be $X \in \mathbb{R}^n$, for each point $X \in \mathcal{M}$, we approximate geodesic distance from X to its neighbors which are located in a ball centered at X such that data is approximately uniformly distributed if creating these custom distance. For each point X, it stretches to a ball centered at X with a fixed radius (the distance from X to its k-th nearest neighbor). We then have a discrete metric space for each point where each edge in the metric space can be weighted by a defined distance function. To merge a family of discrete metric spaces, the natural way is to convert the metric spaces into fuzzy simplicial sets, which provides a combinatorial approach for these discrete metrics. The fuzzy set is defined as follows:

Definition 4. *For an universe of discourse U and $x \in U$, a fuzzy set F is defined as: $F = \{(x, \mu_F), x \in U\}$, where μ_F is a membership function associated with F that maps $x \in U$ to the interval $[0, 1]$. Thus, each element of F is with a fuzzy membership strength.*

The fuzzy simplicial set is a simplicial set (which is made up of simplices) in which every simplex is with a fuzzy membership strength [19]. For a metric space local to X, the distance is considered as the weight (*i.e.,* fuzzy membership strength) of 1-simplices, and we set the distance to the nearest point as 0 while each 0-simplex which is the face of some 1-simplex is defined as 1. When merging two edges which are associated to incompatible metric spaces and with respective weights α and β, the combination of weights together on a single edge is defined as $w(\alpha, \beta) = \alpha + \beta - \alpha\dot{\beta}$ under a probabilistic fuzzy union. Notably, this weight can be referred to as the probability of the existence of the new edge.

6.2.2 *Optimizing based on fuzzy set cross entropy.*

Let $Z \in \mathbb{R}^l$ be a low dimensional representation of $X \in \mathbb{R}^n (l << n)$. To optimize the low dimensional layout of data, we minimize the distance between two fuzzy graphs using cross-entropy: one is formed by the low dimensional set (denoted as $\{Z\}$) and the other is generated by the input set (regarded as $\{X\}$). Assume E is a reference set of all possible 1-simplices, μ_l is the weight of 1-simplex e in low dimensional representation while μ_h is the weight in high dimensional case, then the cross-entropy is defined as:

$$\sum_{e \in E} \mu_h(e) \log\left(\frac{\mu_h(e)}{\mu_l(e)}\right) + (1 - \mu_h(e)) \log\left(\frac{1 - \mu_h(e)}{1 - \mu_l(e)}\right). \quad (1)$$

Here, the first term $\mu_h(e) \log\left(\frac{\mu_h(e)}{\mu_l(e)}\right)$ forces a clump on the points spanned by edges with large weights associated to the high dimensional representation, *i.e.,* $\mu_h(e)$ is high. This is because the first term will be minimized only if $\mu_l(e)$ is large, which indicates the probability of the existence of the simplex is high. Conversely, the last term $(1 - \mu_h(e)) \log\left(\frac{1-\mu_h(e)}{1-\mu_l(e)}\right)$ provides appropriate gaps between the points whose edges associated to high dimensional

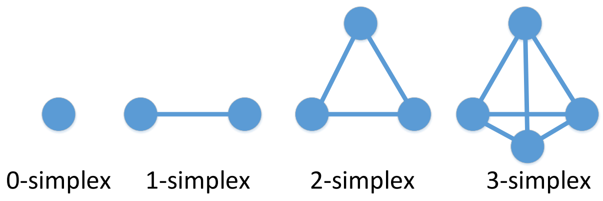


Figure 5: Simplices. 0-simplex is a point, 1-simplex is a line segment, 2-simplex is a triangle (with three 1-simplices as faces), and 3-simplex is a tetrahedron (with four 2-simplices as faces).

case are with small $\mu_h(e)$. This term will be minimized when $\mu_l(e)$ is as small as possible.

When we construct a low-dimensional representation of high dimensional data, it is important to choose a suitable value for the number of neighborhoods of each point. As the number is increased, we manage to capture more global structure of data but some fine details of local structure are lost. Empirically, it is shown that 15-20 is sufficient to cover the overall view of the data. Besides, to save some trouble, we explicitly want the distance to be a Euclidean distance such that the manifold is embedded in low dimensional Euclidean space.

6.3 Auditor

The auditor is defined as a function $au : \mathcal{M}_D \times \mathcal{M}_A \rightarrow [0, 1]$ that determines whether a privacy leakage exists. $\mathcal{M}_D$ is a low-dimensional representation of the training dataset while $\mathcal{M}_A$ represents for the adversaries' dataset. An ideal auditor should hold two properties:

- *Completeness*: if privacy is not preserved due to excessive answers, $Md_t(x) = \emptyset$. In other words, if along with historical $\mathcal{Q}_t$ and $\mathcal{Y}_t$, there exists a privacy leakage if answer the newly received query, the auditor will reject the query.
- *Soundness*: secure queries are never rejected.

An intuitive auditor can be implemented by comparing two individual fuzzy simplicial representations of $\mathcal{D}_t$ and $\mathcal{A}_t$, followed by making use of fuzzy set cross entropy. However, in practical, they are incommensurable regarding cross entropy because these two obtained fuzzy sets are not established on the same reference set F. We thus propose numerical measurements in the following.

6.3.1 *Geometry-based Auditor.*

We first introduce a coarsely numerical measurement by using geometrical approximations. Let $\mathcal{M}_D = \{Z_1, Z_2 \cdots, Z_l\}$ be the set of l-dimensional vector representation for the training set $\mathcal{D}_t = \{X_1, X_2, \cdots, X_n\}$. For computational conveniences, we reconstruct $\mathcal{M}$ and define a spherical manifold, where each sphere is described with a centroid $\mathbf{c} \in \mathbb{R}^l$ and a radius r. Remark that, $\mathcal{M}_D$ will be approximated by a circle if in the case of $l = 2$. We define the centroid and radius as follows:

$$\mathbf{c} = \frac{\sum_i Z_i}{n}, \tag{2}$$

$$r = \frac{||\mathbf{c} - Z_i||_2}{n}. \tag{3}$$

We then have a spherical manifold $\mathcal{B}_D$ (with a centroid $\mathbf{c}_d$ and a radius r_d) defined for $\mathcal{M}_D \in \mathbb{R}^l$. $\mathcal{M}_A \in \mathbb{R}^l$ is described as a sphere manifold $\mathcal{B}_A$ with a radius r_a and located in the centroid $\mathbf{c}_a$. Afterward, we can measure the probability of leakage $\mathbb{P}$ by checking if two spherical manifolds match.

Definition 5. *Given sphere manifold representations $\mathcal{B}_A$ and $\mathcal{B}_D$, for any point $X \in \mathcal{B}_A$, the probability $\mathbb{P}$ is defined as:*

$$\mathbb{P} = P(X_i \in \mathcal{B}_D | X_i \in \mathcal{B}_A). \tag{4}$$

The privacy is preserved if $\mathcal{B}_A$ scarcely intersects with $\mathcal{B}_D$. Otherwise, if $\mathcal{B}_D$ entirely includes $\mathcal{B}_A$ *i.e.,* $\mathbb{P} \geq 1 - \delta$, it is argued that there is a privacy breach. When δ is larger, it encourages our auditor to become much stricter and more relaxed to reject received queries. We will further evaluate this in Section 7. In practice, we generate individual sphere manifolds for examples that belong to the same class in $\mathcal{M}$ such that there will be $|C_t|$ manifolds in total. Thus, $\mathbb{P}$ turns to be $\sum_{c_i}^{C_t} P(X_i \in \mathcal{B}_D^{(c_i)} | X_i \in \mathcal{B}_A^{(c_i)}) |C_t|$.

In the case of $l = 3$, the probability turns to be the volume ratio of the intersection to $\mathcal{B}_A$, *i.e.,* $\frac{Vol(\mathcal{B}_A \cap \mathcal{B}_D)}{Vol(\mathcal{B}_A)}$, the value of which equals to $\frac{\pi(r_d + r_a - d)^2(d^2 + 2dr_a - 3r_a{}^2 + 2dr_d + 6r_a r_d - 3r_d{}^2)}{12d}$, where $d_e = ||\mathbf{c}_D - \mathbf{c}_A||_2$. For $l = 2$, $\mathbb{P}$ is determined by the area ratio.

A shortcoming of this sphere-based auditor is that it fails to capture the details of topology structures. Since we take all points in the reconstructed space to be a single connected blob and neglect the difference between some disconnected and scattered data, the estimated $\mathbb{P}$ may be higher than the ground-truth. An ideal auditor should barely tolerate this.

6.3.2 *Topology-based Auditor.*

To remedy the above issues we propose to use topological approximations inspired by the work of [17]. The difference is that we reconstruct a simplicial complex for the manifold representation $\mathcal{M}$ and then compute the *homology* of corresponding simplicial complexes where homology is a topology property known to be efficiently computable.

The precise definition of homology is defined in [11, 26]. Homology group is generally used to classify topological spaces in algebraic topology. In the case of Fig.6, we have one hole appear in (b) and then disappear in (c) by gradually increasing the value of ϵ. $\epsilon_1 - \epsilon_0$ is then defined as the living time of that hole. It is not sufficient to count the number and the type of holes appearing at each ϵ. The short-lived topological features are always considered as the signal with noise. A barcode is a graphical representation of persistent homology group of chain complexes. It is able to filter out the topological noise and capture significant features of data qualitatively if we use the barcode representation. Building on [11], it is argued that the change of homology with respect to ϵ can be encoded in persistence barcodes. Therefore, we compare the topological properties of $\mathcal{M}_D$ and $\mathcal{M}_A$ based on the barcodes by taking the approach in [17].

The main idea is to measure *the mean of relative living time* (MRLT) of each hole where the relative living time is defined as the ratio of living time to the total time (when all points are connected to a single blob). The topological similarity of two low-dimensional vectors $\mathcal{M}_D$ and $\mathcal{M}_A$ are then measured in the following way:

$$S(\mathcal{M}_D, \mathcal{M}_A) = \sum_i (MRLT(i, \mathcal{M}_D) - MRLT(i, \mathcal{M}_A))^2, \tag{5}$$

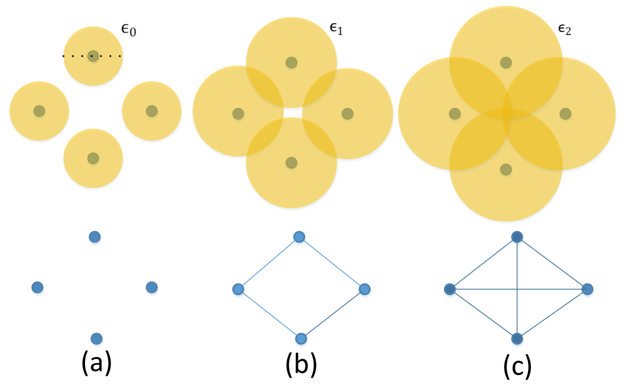


Figure 6: Homology. We have balls with radius ϵ center at each point of $\mathcal{M}$. Then we show different simplicial complexes and homology with increasing values of ϵ. For any subset of size $i+1$ in which all the pairwise balls are mutually intersected, we then add a i-dimensional simplex to the simplicial complex. If the highest dimension of simplices is updated to $k+1$, the so-called k-th rank *homology* H_k is introduced. For example, H_0, H_1 and H_2 are respectively introduced in (a), (b) and (c). There is one one-dimensional hole in (b) while zero two-dimensional hole in (c).

where i represents for the rank of H_i. We first fix the dimension i and then study the corresponding homology.

In our work, we apply fuzzy set cross entropy to optimize low-dimensional representation in two types of auditors: *geometry-based auditor* and *topology-based auditor*. In essence, it is allowed to only apply topology-based auditors by comparing the adversaries' dataset and the training dataset directly in our framework, but the cost resulted from an intolerant delay for high-dimensional homology comparison is an issue. Our evaluation will show the efficiency of our auditor (Section 7).

7 EXPERIMENTS

In this section, we evaluate the accuracy, feasibility, efficiency, and properties described in Section 6. We first describe the experiment setup, followed by the results of our simulators and auditors in several settings. We then study how our defense works against different datasets and different models in detail. Note that all error bars we use in this work represent the standard deviation (SD).

7.1 Experiment Setup

Data. We mainly use MNIST and CIFAR-100 for evaluation due to the distinct accuracy of attacks behaved in [25]: 51% for the first term while the other is up to 97%. On MNIST, there are 60,000 images for the training set and 10,000 examples in the test set. CIFAR-100 have in total 50,000 32×32 colorful training images (there are precisely 5,000 images for each class) and 10,000 test images. We then respectively select different fractions of MNIST and CIFAR-100 to evaluate our defense.

Model. For both MNIST and CIFAR-100, we train a convolutional neural network (CNN) with two convolution and max-pooling layers, followed by a fully connected layer where the hidden neuron size sets to 256. Let Tanh be the activation function, and we set the learning rate to 0.001 and the maximum numbers of the epoch to 100. On MNIST datasets, we additionally train a logistic regression with an L2 penalty and set the batch size to 32, and learning rate to 0.001.

For all our experiments, we use a machine running with Intel(R) Core(TM) i7-6500U CPU, 16GB RAM, and the python libraries such as Theano.

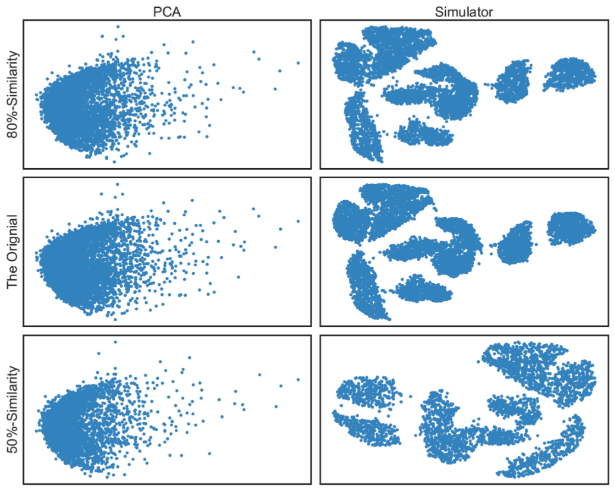


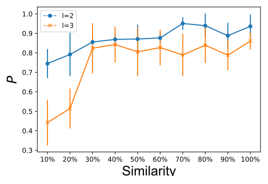
Figure 7: Our simulator vs. PCA. We show manifolds of the MNIST datasets which belongs to the same class (*e.g.,* images of digit one in our case) as an example. The first row shows manifold representations of a dataset which has the similarity 80% with MNIST while the third row has a 50% similarity. For comparative purpose, we put the manifolds of the MNIST dataset in the middle. The manifolds on the left side (generated with PCA) are indistinguishable, which fail to capture sophisticated structures.

7.2 Simulators

We first evaluate the performance of our simulator and compare it with one of the best known manifold learning schemes – PCA. Fig.7 shows the low-dimensional manifolds of different fractions of the MNIST datasets. When we check the three figures on the right side, the manifolds in the first row have better similarity to the middle one compared to the manifolds presented in the third row. Here, similarity means the overlap of two given datasets. The result states that similar distributions likely share the same topology structures. The PCA has a higher efficiency which costs $4s$ on average while our simulator takes up to $172s$. However, the results of Fig.7 exhibit our simulator can capture more sophisticated structures such that it provides an opportunity to further comparisons in the auditor.

7.3 Auditors

Since an auditor is used to compare low-dimensional manifolds drawn from simulators, we thus evaluate our auditors in different


(a) $\mathbb{P}$.

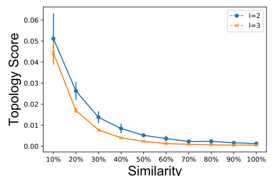

(b) Topology Scores.

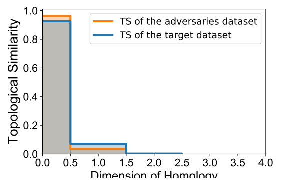
Figure 8: Value of $\mathcal{P}$ and topology scores in the dataset generated from varying fractions of CIFAR-100 (considered to be different similarities).

settings of the dimension (denoted as l). We perform our auditors on comparing the CIFAR-100 dataset and the different numbers of the dataset. Fig.8(a) reports the performance of our geometry-based auditor, and the results highlight the *accuracy* of the auditor that $\mathbb{P}$ approaches to one as the similarity increases (*i.e.,* from 10% to 100%). A striking revelation in Fig.8(a) is that $\mathcal{P}$ is high even for a low similarity when $l = 2$. The reason for the high value on this case is that data of CIFAR-100 concentrates on a low-dimensional manifold such that only a very limited number of data can cover the manifold of the CIFAR-100 dataset. Fig.8(b) reports the topology similarity while varying the size of datasets and l. The results emphasize the *accuracy* of our topology-based auditor: the topology score drops as the similarity increases. However, those values are so minimal that how to define an appropriate threshold in a topology-based auditor remains to be solved. The results show the robustness of our auditors – they have similar outputs whenever $l = 2$ or $l = 3$ if they share the same similarity. From stability account, we set $l = 3$ in the following experiments.

7.4 Feasibility

Surprised by the distinct performances of membership attacks in the MNIST and the CIFAR-100 mentioned before, we recover the experiments by respectively training a CNN model on 80% of the training datasets. We then have 31% test accuracy for CIFAR-100 and 98% for the MNIST. The training dataset of attack model (*i.e.,* adversaries' dataset) is built on the leftover training data such that there is no overlap. We show the topology similarity of adversaries' dataset and the real training dataset in MNIST and CIFAR-100 dataset. A reasonable explanation of the distinct attack performances is presented in Fig.9. The successful attack (which is based on CIFAR-100) is shown to have a topology approximate the real one.

Fig.10(a) and Fig.10(b) report our performances on MNIST. We apply ML defense to a logistic regression model where the test accuracy is 96%. The results indicate that the setting of δ (the privacy-related parameter) should be flexible and need carefully deploying while applied to different datasets on which a model is trained. As depicted in Fig.10(a), if we set δ to 0.2 (*i.e.,* $\mathbb{P}$ should be less than 80%), it allows answering near 30,000 queries in our defense and of which queries are considered the ones with high classification confidence. However, it needs near 110,000 queries on average to get a successful model extraction attack (up to 98%) while the membership attacks with 53% attack accuracy rely on 30,000 synthetic


(a) Topology similarity for MNIST.

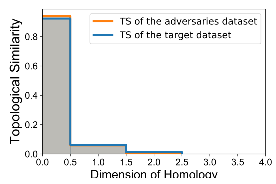
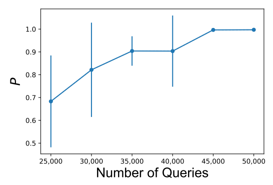

(b) Topology simi. for CIFAR-100.

Figure 9: Topology similarity for images in MNIST/CIFAR-100. We show images belonging to class 1 as a running example. The blue one represents the topology of the data distribution from which the target training data is sampled while the orange one is associated with the training data of the attack models.


(a) $\mathbb{P}$.

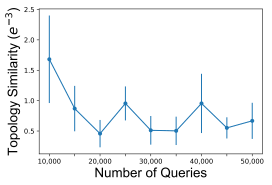

(b) Topology Scores.

Figure 10: ML defense varying numbers of queries in a logistic regression model being trained on MNIST.

records which are generated by $30,000 \times 100$ queries on average. When concerning applicability, a Defensor can calibrate the parameters carefully beforehand and consider the worst case, *e.g.,* start to deny queries when answering 110,000 queries regardless of attack types. These results report that there is a considerable gap between privacy in ML defense and the state-of-art attacks. In addition, our protocol reduces the number of queries being answered (denoted as the *utility* of an ML model). The improvement of utility is left as an open question by our work.

7.5 Efficiency

We apply ML defense to CNN models which are trained on MNIST and CIFAR-100, respectively. Fig.11 and Fig.12 jointly illustrate that the efficiency (one of the most significant properties) of our defense is acceptable and independent of different models. The results emphasize that our geometry-based auditor is efficient. The response time of a query whose serial number is 50,000 (known to be the accumulated number of the query accesses) is nearly 150s while the cost of the topology-based auditor is slightly higher. A delay of 2 minutes per query may be not great in practice when the serial number grows to 50,000, which requires further improvement.

8 DISCUSSION

The effectiveness of ML defense mainly depends on two assumptions: (1) most of the successful prediction API threats fundamentally rely on a similar data distribution derived from excessive query accesses; (2) the low-dimensional manifold representations are comparable based on geometric or topological properties. The empirical evidence exhibit that our assumptions are reasonable. We admit our auditor is so cautious that some queries will be rejected even

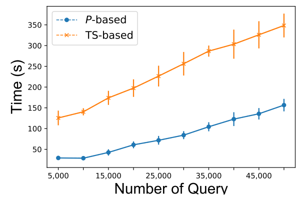


Figure 11: Efficiency. The efficiency of ML defense for a CNN model being trained on the MNIST dataset.

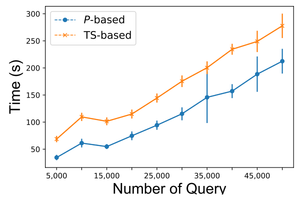


Figure 12: Efficiency. The efficiency of ML defense for a CNN model being trained on the CIFAR-100 dataset.

when they do not lead to privacy breach in the future ML attacks. If the adversaries have multiple accounts to attack the MLaaS, the system can consider all users as a giant adversary and reform the adversaries' dataset based on all query histories. This miscalculation is because defense schemes always have limitations and may lead to query rejections by mistake. We hope that our work can motivate more works to propose a more powerful ML defense such that it can support more queries while taking privacy implications into account.

For the sake of effectiveness, we compare low-dimensional manifold instead of high-dimensional space. The service of quality might be degraded in our protocol since auditor may reject queries even the adversaries' datasets are different from the training set in high-dimension space. We will find more powerful countermeasures to fulfill this gap in the future.

9 CONCLUSION

We have designed and implemented **ML defense**, which is the first framework to defend against prediction API threats. Instead of devising a defense against one specific type of ML attacks, we consequently investigate the common intrinsic properties among the state-of-art ML attacks and propose a general ML defense. Our defense is an independent, quantitative approach such that it does not require to reconstruct either a classifier or a training dataset from the existing MLaaS systems. **ML defense** handles an input query in two steps: finding a low-dimensional representation for adversaries' dataset and then comparing them using either geometric metrics or topology structures. These two procedures jointly provide feasibility and sufficient efficiency of our defense. Experiments demonstrate the effectiveness and efficiency of ML defense.

ML defense is a first step towards finding a general countermeasure to prediction API threats. However, from the utility perspective, we realize that there is an inherent issue of our approach: the accepting ratio will be lower. Thus, finding more powerful auditors is left as an open question. We leave an in-depth study of refining our countermeasures for future work.

ACKNOWLEDGMENTS

X.-Y. Li is the corresponding author. The work is partially supported by the National Key R& D Program of China 2018YFB0803400, China National Funds for Distinguished Young Scientists with No. 61625205, NSFC with No. 61751211, No. 61572347, No. 61520106007, Key Research Program of Frontier Sciences, CAS, No. QYZDY-SSW-JSC002, and NSF CNS-1526638.

REFERENCES

[1] Agrawal, R., and Srikant, R. Privacy-preserving data mining. In *ACM Sigmod Record* (2000), vol. 29, ACM, pp. 439–450.
[2] Ateniese, G., Mancini, L., Spognardi, A., et al. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *IJSN 10*, 3 (2015), 137–150.
[3] Borji, A. Pros and cons of gan evaluation measures. *arXiv preprint arXiv:1802.03446* (2018).
[4] Bost, R., Popa, R., Tu, S., and Goldwasser, S. Machine learning classification over encrypted data. In *NDSS* (2015), vol. 4324, p. 4325.
[5] Cayton, L. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep 12*, 1-17 (2005), 1.
[6] Chen, L., Jung, T., Du, H., Qian, J., Hou, J., and Li, X.-Y. Crowdlearning: Crowded deep learning with data privacy. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)* (2018), IEEE, pp. 1–9.
[7] Cruz, J. A., and Wishart, D. S. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics 2* (2006), 117693510600200030.
[8] Erickson, B., Korfiatis, P., Akkus, Z., and Kline, T. Machine learning for medical imaging. *Radiographics 37*, 2 (2017), 505–515.
[9] Fredrikson, M., Jha, S., and Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS* (2015), ACM, pp. 1322–1333.
[10] Fukunaga, K. *Introduction to statistical pattern recognition.* Elsevier, 2013.
[11] Ghrist, R. Barcodes: the persistent topology of data. *Bulletin of the AMS 45*, 1 (2008), 61–75.
[12] Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML* (2016), pp. 201–210.
[13] Hotelling, H. Analysis of a complex of statistical variables into principal components. *IJEP 24*, 6 (1933), 417.
[14] Hou, J., Li, X.-Y., Jung, T., Wang, Y., and Zheng, D. Castle: Enhancing the utility of inequality query auditing without denial threats. *IEEE Transactions on Information Forensics and Security 13*, 7 (2018), 1656–1669.
[15] Jung, T., Li, X.-Y., Wan, Z., and Wan, M. Privacy preserving cloud data access with multi-authorities. In *2013 Proceedings IEEE INFOCOM* (2013), IEEE, pp. 2625–2633.
[16] Jung, T., Li, X.-Y., Wan, Z., and Wan, M. Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. *IEEE transactions on information forensics and security 10*, 1 (2015), 190–199.
[17] Khrulkov, V., and Oseledets, I. Geometry score: A method for comparing generative adversarial networks. *arXiv preprint arXiv:1802.02664* (2018).
[18] Lin, W.-Y., Hu, Y.-H., and Tsai, C.-F. Machine learning in financial crisis prediction: a survey. *IEEE SMC 42*, 4 (2012), 421–436.
[19] McInnes, L., and Healy, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).
[20] McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. Federated learning of deep networks using model averaging.
[21] Nabar, S. U., Kenthapadi, K., Mishra, N., and Motwani, R. A survey of query auditing techniques for data privacy. In *Privacy-Preserving Data Mining.* Springer, 2008, pp. 415–431.
[22] Nabar, S. U., Marthi, B., Kenthapadi, K., Mishra, N., and Motwani, R. Towards robustness in query auditing. In *Proceedings of the 32nd international conference on Very large data bases* (2006), VLDB Endowment, pp. 151–162.
[23] Narayanan, H., and Mitter, S. Sample complexity of testing the manifold hypothesis. In *Advances in Neural Information Processing Systems* (2010), pp. 1786–1794.
[24] Shokri, R., and Shmatikov, V. Privacy-preserving deep learning. In *CCS* (2015), ACM, pp. 1310–1321.
[25] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *SP* (2017), IEEE, pp. 3–18.
[26] Switzer, R. M. *Algebraic topology-homotopy and homology.* Springer, 2017.
[27] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In *CVRP* (2014), pp. 1701–1708.
[28] Theis, L., Oord, A., and Bethge, M. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844* (2015).
[29] Tramèr, F., Zhang, F., Juels, A., Reiter, M., and Ristenpart, T. Stealing machine learning models via prediction apis. In *USENIX* (2016), pp. 601–618.
[30] Vanhaesebrouck, P., Bellet, A., and Tommasi, M. Decentralized collaborative learning of personalized models over networks. In *AISTATS* (2017).
[31] Zhang, Q., Yang, L. T., and Chen, Z. Privacy preserving deep computation model on cloud for big data feature learning. *IEEE TC 65*, 5 (2016), 1351–1362.