11-2023

Data provenance via differential auditing

Xin MU
Peng Cheng Laboratory

Ming PANG
JD Beijing

Feida ZHU
Singapore Management University, fdzhu@smu.edu.sg

Data Provenance via Differential Auditing

Xin Mu
Peng Cheng Laboratory
Shenzhen, China
mux@pcl.ac.cn

Ming Pang
JD, China
pangm@jd.com

Feida Zhu
Singapore Management University
Singapore
fdzhu@smu.edu.sg

***Abstract*—Auditing Data Provenance (*ADP*), i.e., auditing if a certain piece of data has been used to train a machine learning model, is an important problem in data provenance. The feasibility of the task has been demonstrated by existing auditing techniques, e.g., shadow auditing methods, under certain conditions such as the availability of label information and the knowledge of training protocols for the target model. Unfortunately, both of these conditions are often unavailable in real applications. In this paper, we introduce Data Provenance via Differential Auditing (DPDA), a practical framework for auditing data provenance with a different approach based on statistically significant differentials, i.e., after carefully designed transformation, perturbed input data from the target model's training set would result in much more drastic changes in the output than those from the model's non-training set. This framework allows auditors to distinguish training data from non-training ones without the need of training any shadow models with the help of labeled output data. Furthermore, we propose two effective auditing function implementations, an additive one and a multiplicative one. We report evaluations on real-world data sets demonstrating the effectiveness of our proposed auditing technique.**

I. Introduction

In an era of accelerated digital transformation, data has been widely recognized as an emerging asset class. Data provenance [1], which is to understand where data comes from, how it is collected and how it can be best used, has been assuming ever-increasing importance. One problem in data provenance attracting growing attention from both academia and industry is Auditing Data Provenance (*ADP*), i.e., how to audit if a given piece of data, referred to as *auditing data* [2]–[4], has been used for training a machine learning model.

The problem of ADP distinguishes itself from related research topics such as membership inference attacks with significantly different motivations and solution priorities driven by the growing needs of data-asset-based digital economy. Two prominent examples are (I) Privacy and (II) Incentive governance [5]. Let's examine an example for each scenario: (I) Privacy: Imagine a user's data has been collected and used to train a machine learning model without her knowledge. To protect her data privacy, an objectively rigorous and quantifiable auditing method is necessary to substantiate her challenge in potential disputes. (II) Incentive governance: In a setting where multiple parties each contributes and trades data to collectively train models in a collaborative manner, e.g., federated learning in a decentralized variant in which no central entities are governing incentivization [5]. The incentive allocation in such a scenario would necessarily entail auditing the usage of all parties' data to a sufficiently fine granularity to guarantee trust and fairness.

From a taxonomy point of view, there are two directions for this problem: One direction is based on model-specific techniques. One example along this direction is to directly audit the target model's training process. Techniques, such as regularization and data augmentation which "memorize" information about the training data set in the model, have been proposed without compromising model performance [6]. Unfortunately, most real auditing settings only allow access to the output or the final parameters of the target model, rather than the training process itself. Another example is to directly design a criterion on the model output to compare training data and non-training data with a preset threshold, e.g., the prediction loss [7] or the prediction confidence (e.g., class probability) [8]. Methods along this line suffer from limited generality due to their reliance on specific criteria.

An alternative research direction is based on a shadow training technique, which has demonstrated successful application in auditing deep learning models [2], [3]. The main idea is to use multiple "shadow models" to imitate the behavior of the target model. As the training data for the shadow models are known, the target model can be trained using the labeled outputs of the shadow models. While shadow training technique is promising for a number of scenarios, it raises two technical challenges:

• **Shadow model generation**.The creation of shadow models entails two necessary requirements: (I) The knowledge of the training protocol for the target model; and (II) The generation of training data for shadow models. Requirement (I) is not always guaranteed in real applications. Requirement (II) means it is necessary to generate multiple data sets based on some heuristic rules.

• **Cumulative errors**. The final auditing results depend on multiple intermediate results of machine learning models. It may cause uncertainties in practice as error accumulates.

In this paper, we adopt a different approach by leveraging the following observations: In general, a machine learning model tends to fit the training data well with a relatively high confidence, as, after all, the model has witnessed these data. If we apply a carefully designed function, i.e. auditing function, to transform both the training and non-training data before feeding them as input to the target model respectively, the training data side would result in a much greater difference in

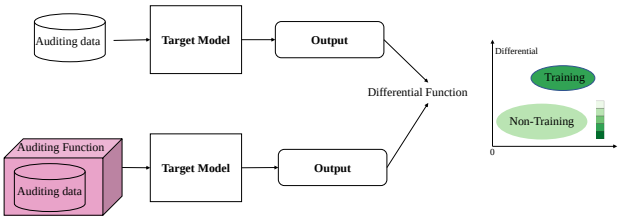


Fig. 1. The DPDA framework.

the target model's output between the original input data and the transformed one. This difference in the confidence of the target model's output between training and non-training data is identified as the key to the DPDA auditing framework we propose in this paper. As illustrated in Figure 1, the auditing framework is constituted by introducing an auditing function, which would be applied to the auditing data before feeding into the target model in one path of the comparison.

Based on this framework, we propose two implementations, an additive implementation (DPDA-ADD) and a multiplicative implementation (DPDA-MUL). DPDA-ADD applies additive transfer functions on input data to generate statistically significant differences, i.e., the changes in model output on training data are more significant than on non-training data. Such additive transfer functions can be easily obtained by maximizing the prediction error of the target model. DPDA-MUL uses a projection function to more carefully emphasize the differential between the training and non-training data, which can be learned by an alternative optimization technique to bridge the auditing model and the target model.

The proposed framework differs from the above mentioned approaches in two key aspects. First, unlike model-specific method, the proposed framework does not rely on specific model output, it can therefore be applied to a wider range of applications to multiple target models. Second, unlike shadow-based method which learns a combination of multiple shadow models to simulate target model, the proposed framework derives a data transformed directly from data through one auditing function, and the mechanism to auditing function is simpler than all these shadow-based approaches.

Another fundamentally important issue re-examined in this paper is the granularity of data that should be the subject of auditing [9]. We argue that what should be used is the notion of *group-based data auditing*, where a set of data points collectively exhibit the characteristics of training or non-training data, because it reflects better the auditing needs of real applications. For example, in applications where a model has been trained with users' facial images or text sets (e.g., tweets), the real question that matters is whether a particular user's data has been used in the training, rather than whether a particular image or tweet of the user has been used. A model should be judged to have already used a user's data if a subset sufficiently characteristic of the user has indeed been used, even if some individual data points are left out in the training.

The contributions of the paper are summarized as follows:

- We present a new framework DPDA for auditing data provenance with a novel notion of differential-based auditing function. Instead of auditing a specific target model in the original input data space, the DPDA framework distinguishes training data from non-training one by comparing a statistical differential generated by the target model between two input data spaces – the original one and the one transformed by an auditing function. Our framework can also adapt quickly to multiple machine learning target models.
- We propose two augmented auditing functions. One is additive implementation DPDA-ADD, the other is multiplicative implementation DPDA-MUL. The proposed DPDA targets the group-based (or user-based) *ADP* problem, such that the auditor can not only infer the membership in a group of data points, but also the membership out of group, delivering a stronger solution than previous ones for the point-based problem as it easily subsumes auditing individual data points.
- We choose three representative benchmark datasets and one real-world application, varying from image to text data sets, to comprehensively evaluate the performance. The effectiveness of our proposed methods have been consistently demonstrated across varied experiment settings. We also study the influence of parameters in the *ADP* problem and provide discussions for some important aspects of our framework for future exploration.

II. Related Work

Membership Inference Attacks (MIAs). The research on auditing data originated from *membership inference attacks* [10], which is to determine if a given data record is in the model's training data set assuming black-box model access [2], [11]. While MIAs has been extensively studied in many fields, e.g., computer vision [7], [8], [12], NLP [3], [9], [13] and recommender system [14]. While MIAs aims to identify training data from attackers' perspective, *ADP* is motivated differently by applications in data-asset-based digital economy [15].

A major line of research is to retrain an inference model to simulate the target model, and then use the inference model to generate multiple results to make final predictions. In [4], authors systematically study the impact of a sophisticated learning-based privacy attacks. When given a differentially private deep model with its associated utility, this paper discusses how much we can infer about the model's training data. Hayes et. al. presented membership inference attacks against Generative Adversarial Networks (GANs) [16]. The idea is that, if a target model overfits the training data, training data will correspond to a higher confidence value on model output. In [3], authors discussed how deep-learning-based text-generation models memorize their training data and provided a solution for text-generation models. These methods are often feasible in settings when certain conditions are satisfied such as the availability of label information or the knowledge of training protocols. However, in many real-life applications, it is difficult , if not impossible, to satisfy these conditions

TABLE I
NOTATION.

Notation	Description
$\mathcal{M}$	Target model
$\mathcal{D}^T$	Training data of $\mathcal{M}$
$\mathcal{D}^O$	Non-training data of $\mathcal{M}$
D_i	A group of data instances
$\|D_i\|$	Size of D_i
$A()$	Auditing function
$\Phi()$	Differential calculation
θ	Target model parameter
W	Auditing function parameter

and hence the severe performance degradation of the auditing mechanisms. More recent work [17] analysed the feasibility of membership inference when the model is overfitted or well-generalized and reported a study that discovered overfitting to be a sufficient but not a necessary condition for data auditing to succeed.

Information Leakage. With the rise of privacy concern for data, many works have been conducted to tackle the problem of information leakage of machine learning model. Information leakage can be grouped mainly into three types: data leakage, model leakage and training environment leakage. For example, in [18], authors demonstrated that embeddings, in addition to encoding generic semantics, often also present a vector that would leak sensitive information about the input data. Deep learning models have been shown to have the ability of memorizing information [19]. Recent work [20] showed that adversaries can extract training text from the output of text generation models, indicating memorization threats to user privacy. Note that this research area can be treated as a direct strategy when the training process is available to auditors.

Differential Privacy (DP) [21] is studied to provide privacy preservation against membership-inference attack in the model inference stage. Many differentially private machine learning algorithms can be grouped according to the basic approaches they use to compute a privacy-preserving model [22]. Some approaches first learn a model on clean data and then use either the exponential mechanism or the Laplacian mechanism to generate a noise model [23], [24]. Some mechanisms add noise to the target function and use the minimum/maximum of the noise function as the output model [25]. It also has been applied to various machine learning models including tree-based model [26], neural networks [27], [28], and federated learning [29], [30]. In this paper, we draw on the idea of *differential* and apply a statistically significant differential for the *ADP* problem.

III. PROBLEM FORMULATION

To best serve the auditing purpose, the granularity of data in this paper that an auditing algorithm is supposed to make judgement upon should be at the group level, where such a group is capable of capturing the characteristics of the underlying entity generating the data. More formally, we associate each entity e with a distribution $\mathcal{A}_e$. A data set D is denoted as $D \leftarrow A$ if D is from distribution A. Two data sets D_i and D_j are said to be *homomorphic under auditing*

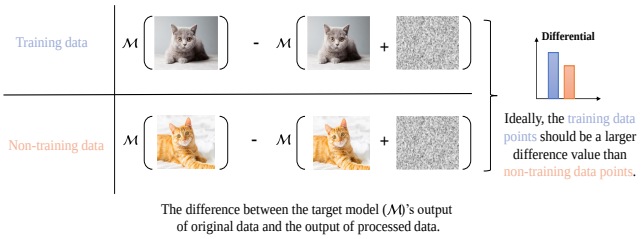


Fig. 2. An illustration of differential mechanism.

if they are both from $\mathcal{A}_e$, denoted as $D_i \cong D_j$. For example, D_i and D_j can be two sets of facial images of the same user e. We show notation in Table I.

We formulate *ADP* as follows:

Definition 3.1: **Auditing Data Provenance (ADP).** Given (I) a set of data distributions $\mathcal{A} = \{A_1, A_2, ..., A_m\}$, (II) a set of groups of data instances $\mathcal{D} = \{D_1, D_2, ..., D_n\}$, such that for each group $D_i \in \mathcal{D}$, $D_i \leftarrow A_k$ for some $1 \leq k \leq m$ and $D_i = \{(x_j, y_j)\}_{j=1}^{|D_i|}$, where $x_j \in \mathbb{R}^d$ is a data instance and $y_j \in Y = \{1, 2, \ldots, c\}$ is its associated class label, and (III) a machine learning model $\mathcal{M}$, which has been trained on $\mathcal{D}^T = \{D_{k_1}, D_{k_2}, ..., D_{k_t}\}$, $\mathcal{D}^T \subset \mathcal{D}$, and its correspondent class probability $P_j = \{p_1, p_2, \ldots, p_c\}$ for each input data instance x_j, the problem of *Auditing Data Provenance (ADP)* is to find a function f such that, for given any auditing data group $D_i \in \mathcal{D}$,

$$f(D_i, \mathcal{M}) = \begin{cases} 1, & \text{if } \exists j, 1 \leq j \leq t, \text{ such that } D_i \cong D_{k_j}, D_{k_j} \in \mathcal{D}^T. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

When $f(D_i, \mathcal{M}) = 1$, we say model $\mathcal{M}$ has used data set D_i for training, denoted as $D_i \in_A \mathcal{D}^T$.

Alternatively, we can define this problem as a ranking problem as follows:

$$\begin{aligned} \max_g \quad & g(D_i, D_j) \\ s.t. \quad & D_i \in_A \mathcal{D}^T, D_j \notin_A \mathcal{D}^T \\ & D_i, D_j \in \mathcal{D},\ 1 \leq i, j \leq n \end{aligned} \tag{2}$$

where $g(\cdot, \cdot)$ represents a similarity function that calculates the difference between two data groups, e.g., Euclidean distance or Cosine distance, etc. The optimization problem is to find a similarity function to maximize the difference between data belonging to the target model $\mathcal{M}$'s training and non-training data. We also notice that it is evident that the group-based *ADP* subsumes auditing individual data points when $|D_i|$ equals 1.

Notice that in general, we can define $\mathbb{O}$ as a distribution distance function such that $\mathbb{O}(A_i, A_j)$ is the distance between any two group distribution, e.g., Kullback-Leibler Divergence or Wasserstein distance. It is hard to discriminate between two data groups if the value of $\mathbb{O}$ is small, namely the two are highly similar. After all, the more similar training and non-training data groups are, the harder auditing problem becomes. In this paper, we also have provided a discussion on the issue of similarity in Section VIII.

In addition, the *ADP* problem is distinguished by two important conditions: the black-box condition and the white-box condition:

• **The black-box condition (BB).** Auditors have no access to $\mathcal{M}$, i.e., no information other than model output can be accessed on model structure or model parameters.

• **The white-box condition (WB).** Auditors have access to all information of $\mathcal{M}$, i.e., model structure, model parameters, and model output, etc.

IV. PROPOSED FRAMEWORK

A. Design Ideas

Our main idea is to propose a **differential mechanism** to distinguish training and non-training data by the output of the target model. The intuition is that training data directly impacts final model parameters. Carefully-designed modification on training data, if successfully transforming training data to one more similar to non-training ones, should result in greater differences in the model output, compared against the differences in model output resulted from the same modification on non-training data as, essentially, little changes have been made in terms of the nature of the input data. As illustrated in Figure 2, simply put, the *differential mechanism* calculates the difference between the target model's output on original auditing data and the output on transformed auditing data. Ideally, training data should generate a larger difference value (represent by blue color) than non-training data.

An extreme yet straightforward case is using **differential mechanism** on instance-based machine learning models, especially those supervised learning models by storing all training instances. [31], [32]. If we take a binary classification model for example, the model must output 1 for training data points, and 0 for non-training data points. It follows that, when given a piece of auditing data, if it indeed belongs to training data, the perturbation added by an auditing function would change the model output from 1 to 0, resulting in a difference of 1. On the other hand, if the auditing data belongs to non-training data, the model output remains 0 after the perturbation, resulting in a difference of 0.

Meanwhile, there are some studies on quantifying the change of the model output by adding a perturbation, e.g., sensitivity analysis [33]–[35].

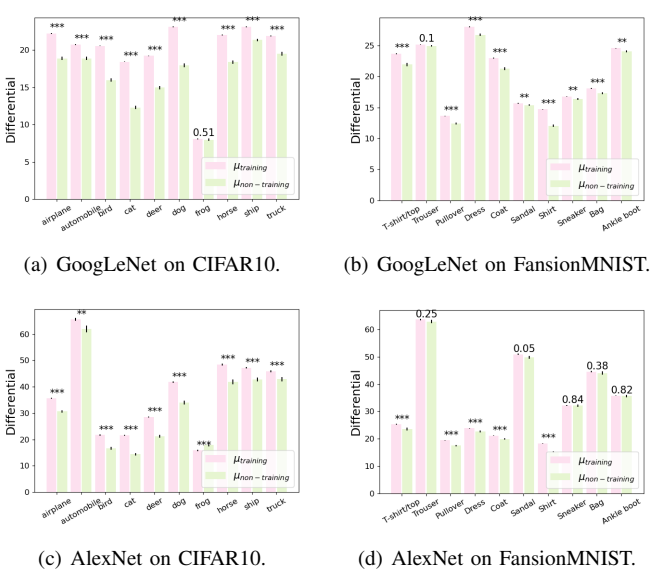
Lemma 1: [33] Consider a Gaussian perturbation $\Delta x \sim \mathcal{N}(0, \varepsilon I)$, the Frobenius norm of the class probabilities Jacobian $||\mathrm{J(x)}||_{\mathrm{F}} = \partial\mathcal{M}/\partial\mathrm{x}^{\mathrm{T}}$, we adopt the Frobenius norm $||\cdot||_{\mathrm{F}}$ estimates the average case sensitivity $\mathcal{M}$ around x:

$$\mathbb{E}_{\Delta x}[||\mathcal{M}(x) - \mathcal{M}(x+\Delta x)||_2^2] \cong \mathbb{E}_{\Delta x}[||\mathrm{J(x)}\Delta\mathrm{x}||_2^2] = \varepsilon||\mathrm{J(x)}||_{\mathrm{F}}^2.$$

Lemma 2: [35] To link the loss function to the output's sensitivity to its input, a first order Taylor expansion can be used to show the sensitivity:

$$\mathcal{M}(x+\Delta x) - \mathcal{M}(x) \cong \Delta x \cdot \nabla_x^T \mathcal{M}(x).$$

Shu et. al. [34] compared network sensitivity between training and testing sets, and demonstrated the existence of difference between the two sets. They contribute to the justifiability of the **differential mechanism** underlying *ADP* to consider difference between training and non-training data by the output of the target model.



(a) GoogLeNet on CIFAR10. (b) GoogLeNet on FansionMNIST.

(c) AlexNet on CIFAR10. (d) AlexNet on FansionMNIST.

Fig. 3. The results of differential mechanism. The Y-axis is a difference value between the output of processed data point and original data point. We use average result of all training or non-training data points per class. We check 95% confidence intervals, and report the p-values obtained via t-tests to assess the statistical significance of differences between the average results of all training and non-training data points. $*: p < 0.05$; $**: p < 0.01$; $***: p < 0.001$.

To further verify the **differential mechanism**, we examine two popular deep learning structures: GoogLeNet [36] and AlexNet [37] on FashionMNIST and CIFAR-10 to observe the effectiveness of our proposed differential mechanism. The experiment is as follows: we select two popular deep learning structures: GoogLeNet and AlexNet on FashionMNIST and CIFAR-10 as observed experiments. Firstly, all auditing data (both training data and non-training data) are processed by adding Gaussian noise. Secondly, we calculate the difference value per class between the output of processed auditing data points and original auditing data points and report the average results. Note that this calculation has been done on training and non-training data separately. $p - value$ shows that there exists a gap of statistical significance between the differences of model output on training and non-training data. The results in Figure 3 show that training data corresponds to a larger difference value than non-training data in most cases. This observation demonstrates the effectiveness of our proposed differential mechanism.

B. DPDA

Based on the differential mechanism concept, we propose a data provenance framework via differential auditing (DPDA), and introduce an auditing function to implement the differential mechanism as discussed in the previous section. As shown in Figure 1, DPDA comprises two main steps: (I) Auditing data is first processed by the auditing function; (II) The difference is calculated for the target model's output between original data and processed data. Finally, the difference is

evaluated to decide whether or not the auditing data belongs to training data.

In DPDA, the auditing function is chosen as a mathematical function formally defined as follows:

Definition 4.1: **Auditing Function (AF)**: Given an auditing data point $x \in \mathbb{R}^d$, the auditing function, denoted as $A()$, is defined as a bijective function such that $A(x) \in \mathbb{R}^d$ and $\forall x, x' \in \mathbb{R}^d, A(x) = A(x') \mapsto x = x'$.

For example, if $A()$ is an additive transformation, we have $A(x) = x + \eta$, $\eta \in \mathbb{R}^d$.

Auditing Function Design. The key to DPDA is to design an auditing function to embed the auditing data into a new space such that it maximizes the differential between training and non-training data, and characterizes the relation between the task of auditing data and the original task of target model. In this paper, we propose three auditing function design as follows. It should be noted that the choices of auditing function design are not limited to these.

(1) **Offset Form.** Offset form is the most common way to do data transformation. Considering a data point $x \in \mathbb{R}^d$, an offset $z \in \mathbb{R}^d$ and a scale $\beta \in \mathbb{R}$, an auditing function in offset form is represented by $A(x) = \beta x + z$.

(2) **Projection Form.** Given a data point $x \in \mathbb{R}^d$ and a matrix $V \in \mathbb{R}^{d \times d}$, the projection form is defined as $A(x) = Vx$.

(3) **Non-linear Form.** Non-linear transfer is widely used in machine learning algorithm design, like tree-based data transfer model [38] or activation function [39].

Note that while (1) and (2) have the advantage of being easy to interpret and fast to use, non-linear forms, on the other hand, are more capable to model real-world data in many cases, due to the greater complexity. In the following, we explore two implementations under this framework: one additive and one multiplicative.

V. Additive implementation

We first present an additive auditing function implementation by a simple offset method as follows:

$$A(x) = x + \varepsilon\eta \tag{3}$$

where ε is a slack variable and η is an offset. Note that the purpose of introducing η is to generate a larger difference between training data and non-training data.

In general, as the target model has seen the training data, its output on training data should have higher confidence, i.e., it should be able to correctly predict data points from training data with high probability. Consequently, for an auditing data point x that is in the training data, if we can induce the processed input $A(x)$ to be *misclassified*, the target model output on x and $A(x)$ should then be more likely to generate a larger difference than the case if x is from non-training data. We would now consider how to induce misclassification on processed data points $A(x)$.

To that end, we review the adversarial example learning perspective [40]. An adversarial example is a widely-used way to conduct an attack. Attackers alter inputs by adding small, often imperceptible, perturbations to force a learned classifier to misclassify the resultant adversarial inputs, which would still be correctly classified by a human observer [41], [42]. Goodfellow et.al. [43] provided a strategy to use the linear view to generate adversarial inputs. Let θ be the parameters of a model, x be the input to the model, y be the label associated with x, $J(\theta)$ be the cost to train the model, ε is a slack variable and the adversarial example can be defined by

$$x^{'} = x + \varepsilon \cdot sign[\nabla_x J(\theta)] \tag{4}$$

Following this idea, we set η in Eqn. (3) as the sign of the target model's cost function gradient

$$\eta = sign[\nabla_x J(\theta)]. \tag{5}$$

This setting would maximize the loss function and result in the greatest misclassification for the auditing data processed by $A()$.

Algorithm 1 Additive Implementation

Input: $D = \{D_1, D_2, \ldots, D_e\}$ - auditing data, $\mathcal{M}$ - target model
Output: D_t - training data

1: **if** $\mathcal{M}$ is not available **then**
2: $O \leftarrow \mathcal{M}(D)$, # Calculate the output.
3: $\mathcal{M}_r \leftarrow$ train a model on $\{D, O\}$
4: **end if**
5: **for** $i = 1, ..., e$ **do**
6: $D_i' \leftarrow A(D_i)$, # using Eqn.(3)
7: $S_i \leftarrow \Phi\langle \mathcal{M}(D_i'), \mathcal{M}(D_i)\rangle$
8: **end for**
9: **if** $\forall_j \ \ S_j > threshold$ **then**
10: $D_j \leftarrow$ training data
11: **else**
12: $D_j \leftarrow$ non-training data
13: **end if**

Algorithm 1 illustrates the sketch of the additive implementation. Lines 5-13 show the case when the target model is under the white-box assumption. When the target model is under the black-box assumption, we need to build an extra machine learning model to imitate the prediction behaviors of the target model. We employ a simulation model trained by auditing data and its model output (e.g., SVM-based model). Then we use this simulation model to calculate η. The process is described in Algorithm 1 Lines 1-4.

Threshold determination. In Algorithm 1 Line 9, a threshold is needed in Algorithm 1 to decide whether the auditing data belongs to training data. As mentioned in previous discussions, we expected that training data have a larger value of differentials than non-training data, that is to say, $\{S_1, S_2, \ldots, S_e\}$ should form two groups in distributions. We adopt the following method [44] to identify the best threshold to separate the two groups. We first generate a list Q of all values in $\{S_1, S_2, \ldots, S_e\}$ in descending order. A threshold τ in this list yields two sub-lists, Q^l and Q^r respectively as the left sub-list and the right sub-list. The following criterion minimises the difference in standard deviations $\sigma(\cdot)$:

$\hat{\tau} = \arg\min_{\tau} \ |\sigma(Q^r) - \sigma(Q^l)|$ The threshold $\hat{\tau}$ is used to differentiate between training and non-training data, the former corresponding to larger values and the latter smaller ones. The details are provided in the Algorithm 2 and an example are showed in the section VII-B.

Algorithm 2 Determining Threshold

Input: Q - the list of value s in $\mathcal{B}$, m - size of Q, $\tau^\star$ - initialize to a larger value. $\sigma(\cdot)$ - standard deviations calculation
Output: $t^\star$ - threshold

```
for i = 1, ..., m do
    Q_l ← Q[1 : i]
    Q_r ← Q[i : m]
    τ ← |σ(Q[1 : i]) − σ(Q[i : m])|
    if τ < τ* then
        t* ← Q[i]
        τ* ← τ
    end if
end for
```

VI. Multiplicative Implementation

In this section, we introduce the multiplicative auditing functions. We have shown in Eqn. (2) that the *ADP* problem can be treated as a ranking problem. It follows that it can be turned into a differential optimization problem, and we can search for auditing function $A()$ by the following objective function:

$$\max \quad \Phi\left\langle \mathcal{M}(A(D_t)), \mathcal{M}(D_t)\right\rangle - \Phi\left\langle \mathcal{M}(A(D_o)), \mathcal{M}(D_o)\right\rangle \quad (6)$$

where $\mathcal{M}(\cdot)$ is the target model output, $A(\cdot)$ is the auditing function, D_t is $\mathcal{M}$'s training data, D_o is the non-training data and $\Phi(\cdot,\cdot)$ represents the differential calculation function. Since Eqn. (6) would aim for the maximum difference, the training and non-training data would therefore exhibit significant gaps for the differential results.

Specifically in this work, we define a multiplicative implementation by a projection as follows:

$$A(x) = Wx, \quad (7)$$

where $W \in \mathbb{R}^{d\times d}$. Thus, Eqn. (6) can be transformed to the following problem:

$$\max \quad \Phi\left\langle \mathcal{M}(W(D_t)), \mathcal{M}(D_t)\right\rangle - \Phi\left\langle \mathcal{M}(W(D_o)), \mathcal{M}(D_o)\right\rangle \quad (8)$$

Optimization. It is important to note that Eqn. (8) requires optimization on D_t and W simultaneously. We employ an alternating optimization algorithm to solve it. We initialize labeled auditing data for learning W by the following steps: Randomly initialize W, and calculate the value Φ of auditing data D. Then calculate a threshold (as mentioned in the section V) to separate the value Φ, and label training data D_t and training data D_o. The optimization procedure is as follows:

(1) We consider W as a variable. D_t and D_o are set as described above. The gradient descent technique is then applied to efficiently solve Eqn. (8).

(2) After W is obtained, we calculate the value Φ of D.

Algorithm 3 Multiplicative Implementation

Input: $\{D_1, D_2, \ldots, D_e\}$ - auditing data, $\mathcal{M}$ - target model, D_c - initialization of training data.
Output: D_t - training data

```
repeat
    calculate W by using Eqn. (9)
    S_i ← {Φ⟨M(WD_i), M(D_i)⟩}_{i=1}^e
    if ∀_j  S_j > threshold  then
        D_j ← training data
    else
        D_j ← non-training data
    end if
until Maximum number of iterations.
```

(3) Calculate the threshold (Algorithm 2) to separate the value Φ of auditing dataset D. The larger ones are set as D_t, the others as D_o.

(4) Use the newly updated D_t and D_o to calculate W.

The procedure stops when the terminating condition is satisfied, i.e., a predetermined maximum number of iterations.

Note that we can apply the gradient descent algorithm to efficiently update W as follows:

$$W' = W - \epsilon\frac{\partial J(W)}{\partial W} \quad (9)$$

where

$$\begin{aligned}\frac{\partial J(W)}{\partial W} =&[\mathcal{M}(WD_t) - \mathcal{M}(D_t)]\mathcal{M}'(WD_t)D_t \\ &- [\mathcal{M}(WD_o) - \mathcal{M}(D_o)]\mathcal{M}'(WD_o)D_o\end{aligned} \quad (10)$$

The sketch of the process is described in Algorithm 3. Note that when $\mathcal{M}'$ is not available under the black box assumption, a simulation model can be employed.

VII. Experiment

A. Experiment Setup

Data Sets. We use four datasets to compare the performance of all methods: **MNIST**, **20 Newsgroups**, **CIFAR-10** and **VGGFace**.

Competing Algorithms. A brief description of each of the methods used in the experiment is given as follows

(1) **Confidence Criteria (CC)**: CC means we directly design a criterion on the model output (e.g., the perdition class probability). The criterion is like a preset threshold, the data with higher probability is treated as training data.

(2) **Shadow Learning Technique (SLT)** [2]: SLT introduces multiple shadow models and an attack model to address the data auditing problem. The shadow model is to recognize differences in the target model's predictions on the inputs that it has trained on versus the inputs that it has not trained on. The attack model is treated as a classifier to distinguish the output of shadow model.

(3) **DPDA-RN** (RN for short): DPDA with the additive implementation but by setting random values.

(4) **DPDA-ADD** (ADD for short): DPDA with the additive implementation.

TABLE II
RESULTS OF DIFFERENT TARGET MODELS ON DIFFERENT DATA SETS.

		MNIST		20 Newsgroups		CIFAR10	
	Algorithm	F-measure	AUC	F-measure	AUC	F-measure	AUC
SVM-based model	CC	0.412 ± 0.02	0.513 ± 0.01	0.401 ± 0.02	0.535 ± 0.01	0.534 ± 0.02	0.535 ± 0.01
	SLT	0.612 ± 0.05	0.805 ± 0.04	0.715 ± 0.03	0.711 ± 0.05	0.704 ± 0.05	0.750 ± 0.03
	RN	0.536 ± 0.03	0.550 ± 0.02	0.433 ± 0.02	0.565 ± 0.02	0.586 ± 0.03	0.526 ± 0.02
	ADD	0.695 ± 0.04	0.822 ± 0.06	0.660± 0.02	0.735 ± 0.02	0.700 ± 0.02	0.803 ± 0.04
	MUL	0.719 ± 0.06	0.821 ± 0.05	0.723 ± 0.02	0.750 ± 0.02	0.726 ± 0.02	0.801 ± 0.04
Tree-based model	CC	0.423 ± 0.02	0.533 ± 0.01	0.339 ± 0.01	0.554 ± 0.02	0.540 ± 0.04	0.573 ± 0.04
	SLT	0.696 ± 0.03	0.711 ± 0.04	0.655 ± 0.02	0.684 ± 0.02	0.744 ± 0.02	0.753 ± 0.05
	RN	0.671 ± 0.05	0.654 ± 0.06	0.632 ± 0.06	0.652 ± 0.06	0.651 ± 0.07	0.673 ± 0.04
	ADD	0.675 ± 0.03	0.652 ± 0.05	0.653 ± 0.03	0.654 ± 0.02	0.675 ± 0.02	0.654 ± 0.05
	MUL	0.695 ± 0.01	0.712 ± 0.04	0.666 ± 0.02	0.704 ± 0.03	0.760 ± 0.03	0.772 ± 0.02
NN-based model	CC	0.493 ± 0.01	0.565 ± 0.03	0.432 ± 0.02	0.515 ± 0.02	0.478 ± 0.02	0.523 ± 0.01
	SLT	0.743 ± 0.03	0.811 ± 0.02	0.693 ± 0.01	0.701 ± 0.01	0.721 ± 0.02	0.798 ± 0.03
	RN	0.521 ± 0.01	0.554 ± 0.02	0.442 ± 0.01	0.542 ± 0.01	0.571 ± 0.01	0.563 ± 0.01
	ADD	0.739 ± 0.04	0.792 ± 0.02	0.703 ± 0.03	0.724 ± 0.02	0.726 ± 0.04	0.803 ± 0.03
	MUL	0.782 ± 0.03	0.802 ± 0.01	0.723 ± 0.02	0.751 ± 0.02	0.719 ± 0.01	0.810 ± 0.04
ADD and MUL have #wins/#draws/#losses		3/3/0		6/0/0		5/1/0	

(5) **DPDA-MUL** (MUL for short): DPDA with the multiplicative implementation.

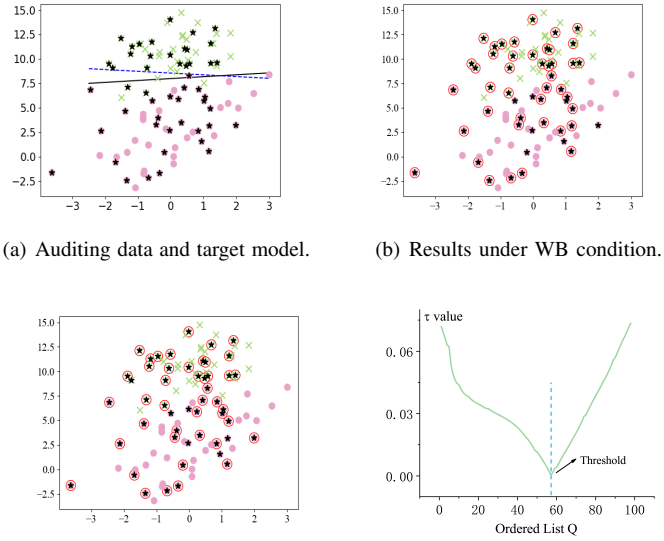
Experiment Settings. All experiments are implemented in Python on Intel Core CPU machine with 128 GB memory and NVIDIA RTX 3090 GPU. The following implementations are used: In CC, the confidence of one instance is set by the largest value of its estimated label probability. In SLT, the codes are developed based on the original paper[1]. We employ 50 shadow model and one attack model which is SVM[2] with RBF kernel and other parameters are set by default values. In DPDA-RN, random values are set by Gaussian noise. The parameter ε in DPDA-ADD is set by $[e^{-8}, e^{8}]$. In particular, we can use data similarity as a measure to guide the setup: the higher the data similarity, the smaller the value ε. In DPDA-MUL, W is initialized to a semi-positive definite matrix.

Evaluation Metrics. We use **F-measure** and **AUC**[3] to measure performance. As mentioned in the definition of *ADP*, auditing data are in the form of groups. We conduct both AUC and F-measure on group-based dataset as follows: First, each auditing data is fed as input for prediction, and we calculate the average results of each group. Then, the AUC or F-measure results are calculated in those group results. Note that each group contains the same type of data, i.e., either all training data or all non-training data.

B. Results on synthetic data

We take SVM-based, Tree-based and NN-based model as the target model and train them on a two-dimensional synthetic dataset with 100 data points. Figure 4 shows SVM-based model results. We indicate two kinds of data in green and red respectively, and mark training data by "black star". The auditing problem is to identify the "black star" training data given all data points as the input auditing data.

Figure 4(a) shows the target model by a solid black line and a simulation model by blue dotted lines. The effectiveness of



(a) Auditing data and target model. (b) Results under WB condition. (c) Results under BB condition. (d) Threshold setting.

Fig. 4. An illustration on SVM-based model.

the proposed method is demonstrated by the auditing results as labeled with red circles in Figure 4, in which (b) shows the white-box condition and (c) shows the black-box condition. In addition, we show the threshold setting in Figure 4(d). The 100 data points in the synthetic data set are divided into training and non-training groups with the ratio of 1:1. Figure 4(d) shows an example of the distribution for τ ($= |\sigma(Q^r) - \sigma(Q^l)|$) curve. Note that the lowest point provides a clear guide to separate the auditing data into the two parts of training and non-training, and it is close to the optimal value of 50.

C. Results on benchmark datasets

Setting. Each dataset is used to simulate the following auditing environment. We first randomly select two classes, and instances of these two classes are selected as an auditing data set, which are then randomly divided into training data and non-training data with the ratio between them being roughly

[1]https://github.com/spring-epfl/mia
[2]https://scikit-learn.org/stable/modules/svm.html
[3]https://en.wikipedia.org/wiki/Receiver_operating_characteristic

1:1. The training set is then used to train a target model to be audited. Subsequently the auditing data are grouped into multiple subsets, each containing the same type of data, i.e., all training data or all non-training data. All experiments are under the BB assumption. Target models are SVM-based models, Tree-based models and NN-based models respectively: SVM-based models are set by a least squares SVM classifier; Tree-based models use random forest with completely random trees; NN-based models are set by two fully connected layers and a SoftMax layer. Parameters of target models are set by default according to their official code package. We run 30 independent experiments with different simulations on each dataset.

Summary. Table II provides more comprehensive results on SVM-based, Tree-based and NN-based target model. Our proposed DPDA models, both DPDA-ADD and DPDA-MUL, have produced higher AUC performance in all data sets than all other methods. The closest contender SLT, which is based on shadow model technique, is weaker than DPDA. DPDA-RN is based on random perturbation and its performance falls behind DPDA in all data sets. The performance of CC ranks at the bottom. An analysis is provided below:

• CC performs worse than others in all data sets. This shows that efforts to directly use the prediction probability are unsuccessful for the *ADP* problem. There are a couple of reasons for this, e.g., the distribution of the model output could be dense and therefore makes the separation of training and non-training data difficult. It is thus concluded that CC is not a good choice for this task.

• SLT requires training multiple shadow models and an attack classification model. It is important to note that its performance highly dependents on shadow model results. Unsatisfactory results of shadow model will severely limit the attack model's ability for classification. In addition, SLT needs label information for training shadow models, which is often hard to obtain in real applications. Nevertheless, the experimental results show that it still performs worse than the DPDA framework in two out of three data sets.

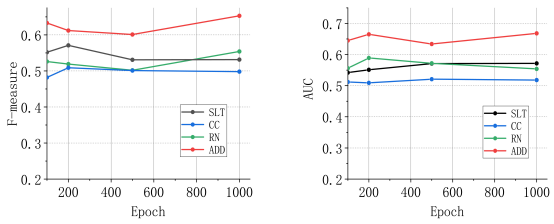
• RN presents worse performance than other DPDA models except on the tree-based model. The under-performance of auditing functions of random values drives home the effectiveness of the two augmentations we proposed for auditing functions.

• Both ADD and MUL are demonstrated to be competitive methods for the *ADP* problem. While MUL achieves a higher performance, ADD excels with its lower computational cost in an extensive parameter search and easier implementation. The choice between them in real applications should be a result of comprehensive consideration on case-dependent factors.

D. Results on a real-world application

Gender estimation is an important and challenging task in many real-world applications. Over the past few years, most methods used deep learning models to estimate gender achieved respectable results [45]–[47]. In this section, we evaluate DPDA and contenders on auditing a gender estimation model. The gender estimation model is set by a famous computer vision machine learning model, ResNet18 [48], including 18 layers deep neural network. And it is trained on VGG-face dataset [49]. The aims of this section are to examine the ability of DPDA to (i) adapt to a real-world group-based auditing problem; and (ii) achieve a good performance.

Fig. 5. The illustration of the gender estimation dataset.

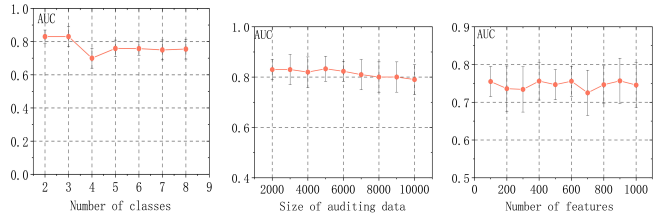


(a) F-measure. (b) AUC.

Fig. 6. The results on auditing a gender classifier.

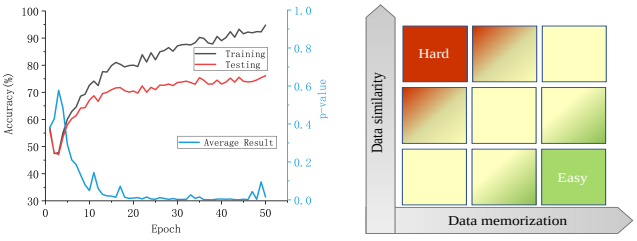
Setting. In the experiment, we used 20 people with 20k images as training data to train a gender estimation classifier. The *auditing data* consist of these 20 people's images which include some training images and some other images which haven't been used to train. In the auditing data, images of each people are naturally regarded as a group D_i. As an example shown in Figure 5, person 1 and person 2 have been used to train the target model, their images are shown in Figure 5. In the auditing data, the images from person 1 and person 2 are both treated as training, even some images haven't been used. Due to the data of person 3 without participating training target model, the images of person 3 are treated as non-training. This is a real application under the group-based assumption. All experiments are under the black-box assumption. Parameters of target models are set by default according to their official code package.

Summary. Figure 6 shows the auditing results over different epochs. In the different epochs, target model ResNet18 has different performance, namely accuracy is [0.64,0.78,0.79,0.82] on [100,200,500,1000]. Firstly, because the proposed method DPDA takes into account the global auditing results in a group, avoiding the effect of outliers, ADD performs better results than all three methods under the group-based auditing assumption. Secondly, SLT is a point-based auditing algorithm which requires to have multiple shadow models in order to simulate the target model. Despite this advantage, it still performed worse than ADD. Meanwhile, the other two baselines also perform worse than ADD. Overall, the experimental results



(a) Different sizes of classes (b) Different sizes of data. (c) Different numbers of features of auditing data.

Fig. 7. Parameter analysis result.



(a) (b)

Fig. 8. (a) The differential mechanism on different epochs. (b) The relation of data similarity, data memorization and ADP problem.

here demonstrate the reasonableness and feasibility of this method on the group-based auditing problem.

E. Parameter analysis

We present a study of parameters in DPDA, i.e., the number of classes in auditing data set, the size of auditing data set and the number of features of auditing data. We evaluate them one at a time on varied settings with other parameters fixed.

Figure 7(a) describes the number of different sizes of classes in auditing data set and Figure 7(b) describes the results of the sizes of auditing data set. We show the results of MUL on MNIST data set. Note that similar results are also observed under the ADD implementation. There are a downward trend for performance as the number of classes and the size of data increase. That said, the denser data distribution may yield a higher degree of data similarity, and make it significantly more difficult to audit data. Figure 7(c) shows the number of features of auditing data. We test varied sizes of the feature vector on the 20 Newsgroups data set. Results show that the different sizes of feature vector have a relatively small impact on performance.

VIII. Discussion

(1) **Auditing Data Distribution.** One challenge of the auditing problem is how to effectively handle the situation when there exists a high degree of similarity between training and non-training data points. Intuitively, it is hard to discriminate training and non-training data points if the two are highly similar. Table III depicts the relation between data similarity and differential mechanism. The table shows results on CIFAR10 data set with the target model set as GoogleNet. For each class, we calculate the average pair-wise Euclidean distance between training and non-training data points, i.e., the smaller the distance value, the more similar the two data sets. For class 'frog', there is a high degree of similarity between training and non-training data points (as shown with the lowest value of 54.21). In this case, while the accuracy achieves 80%, the differential mechanism generates the worst result, i.e., it corresponds to the largest p-value. These data indicate that the high degree of similarity poses a big challenge to DPDA.

We put forward a plan to address this issue as our future work – auditing functions can be treated as a transfer function in that it shifts the original data space to a new space where it is easier to solve the problem of the high degree of similarity. Alternatively, a method like Generative Adversarial Networks [50] can be used to achieve this transfer, which offers a better capture and understanding of the distribution of training and non-training data.

(2) **Data Memorization.** Data memorization refers to a model's capacity to remember its input data, especially its training data [51]. In DPDA, data memorization of the target model is another important factor for the effectiveness of the differential mechanism. Figure 8 (a) plots the differential mechanism along different training epochs. It can be observed that the performance of target model is often poor due to under-fitting in early epochs of model training. It is fair to say that the target model has poor memorization on training data during this period. It follows that the difference between the model output of training and non-training is not sufficiently significant, resulting in larger p-values. However, it can be clearly observed that, as the number of epochs increases, the target model's memorization of training data gets better and better, resulting in smaller and smaller p-values. This demonstrates that, the better the data memorization of the target model, the more effective the proposed differential mechanism for the *ADP* problem.

In conclusion, we describe the relation of data similarity, target model memorization and differential mechanism in 8 (b). In face of data with high similarity and model with poor memorization, it is hard for the differential mechanism to handle the *ADP* problem. In contrast, for data with low similarity and model with good memorization, the differential mechanism would work well.

IX. Conclusions

This paper investigates an important problem in data provenance, i.e., algorithmically check if a piece of data has been used to train a machine learning model. We introduce a new auditing framework DPDA based on the idea of differential and propose two implementations of the auditing function, additive implementation and multiplicative implementation. Extensive experiments on real-world data sets have demonstrated the effectiveness of both the proposed methods. We provide discussions for some important aspects of our framework in the ADP problem.

References

[1] P. Buneman, S. Khanna, and W. C. Tan, "Why and where: A characterization of data provenance," in *ICDT*, 2001, pp. 316–330.

TABLE III

THE RELATION BETWEEN DATA SIMILARITY AND DIFFERENTIAL MECHANISM. (DATA SIMILARITY IS CALCULATED BY AVERAGE PAIR-WISE EUCLIDEAN DISTANCE BETWEEN TRAINING AND NON-TRAINING DATA, I.E., THE SMALLER THE VALUE, THE HIGHER THE SIMILARITY. THE MEANING OF P-VALUE IS THE SAME AS FIGURE 3.)

	'car'	'cat'	'truck'	'dog'	'horse'	'plane'	'ship'	'deer'	'bird'	'frog'
Data similarity	65.05	63.73	62.44	61.86	60.99	58.83	57.207	55.78	54.87	**54.21**
Accuracy	63%	54%	69%	52%	69%	65%	70%	72%	75%	**80%**
p-value	$***$	$***$	$***$	$***$	$***$	$***$	$***$	$***$	$***$	**0.507**

[2] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *S&P*, 2017, pp. 3–18.

[3] C. Song and V. Shmatikov, "Auditing data provenance in text-generation models," in *KDD*, 2019, pp. 196–206.

[4] M. A. Rahman, T. Rahman, R. Laganière, and N. Mohammed, "Membership inference attack against differentially private deep learning model," *Transactions on Data Privacy*, vol. 11, no. 1, pp. 61–79, 2018.

[5] Q. Yang, *Federated Learning: Privacy and Incentive*. Springer Nature, 2020.

[6] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *CCS*, 2017, pp. 587–601.

[7] 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 *S&P*, 2019, pp. 739–753.

[8] 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 *NDSS*, 2019.

[9] V. Shejwalkar, H. A. Inan, A. Houmansadr, and R. Sim, "Membership inference attacks against NLP classification models," in *NeurIPS 2021 Workshop Privacy in Machine Learning*, 2021.

[10] H. Hu, Z. Salcic, G. Dobbie, and X. Zhang, "Membership inference attacks on machine learning: A survey," *CoRR*, vol. abs/2103.07853, 2021.

[11] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Towards demystifying membership inference attacks," *CoRR*, vol. abs/1807.09173, 2018.

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

[13] X. Pan, M. Zhang, S. Ji, and M. Yang, "Privacy risks of general-purpose language models," in *S&P*, 2020, pp. 1314–1331.

[14] M. Zhang, Z. Ren, Z. Wang, P. Ren, Z. Chen, P. Hu, and Y. Zhang, "Membership inference attacks against recommender systems," in *CCS*, 2021, p. 864–879.

[15] J. Pei, "Data pricing - from economics to data science," in *KDD*, 2020, pp. 3553–3554.

[16] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro, "LOGAN: membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 133–152, 2019.

[17] Y. Long, V. Bindschaedler, L. Wang, D. Bu, X. Wang, H. Tang, C. A. Gunter, and K. Chen, "Understanding membership inferences on well-generalized learning models," *CoRR*, vol. abs/1802.04889, 2018.

[18] C. Song and A. Raghunathan, "Information leakage in embedding models," in *CCS*, 2020, pp. 377–390.

[19] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *ICLR*, 2017.

[20] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *USENIX Association*, 2019, pp. 267–284.

[21] C. Dwork, "Differential privacy," in *ICALP*, 2006.

[22] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: a survey and review," *CoRR*, vol. abs/1412.7584, 2014.

[23] J. Vaidya, B. Shafiq, A. Basu, and Y. Hong, "Differentially private naive bayes classification," in *ICWI*, 2013, pp. 571–576.

[24] K. Chaudhuri, A. D. Sarwate, and K. Sinha, "Near-optimal differentially private principal components," in *NIPS*, 2012, pp. 998–1006.

[25] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft, "Learning in a large function space: Privacy-preserving mechanisms for SVM learning," *Journal of Privacy and Confidentiality*, vol. 4, no. 1, 2012.

[26] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A practical differentially private random decision tree classifier," in *ICDM Workshops*, 2009, pp. 114–121.

[27] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016, pp. 308–318.

[28] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *CCS*, 2015, pp. 1310–1321.

[29] P. Kairouz, H. B. McMahan *et al.*, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019.

[30] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, vol. abs/1712.07557, 2017.

[31] R. S. Michalski and J. R. Anderson, *Machine learning - an artificial intelligence approach*, ser. Symbolic computation. Springer, 1984.

[32] D. W. Aha, D. F. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[33] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: an empirical study," in *ICLR*, 2018.

[34] H. Shu and H. Zhu, "Sensitivity analysis of deep neural networks," in *AAAI*, 2019, pp. 4943–4950.

[35] M. Forouzesh, F. Salehi, and P. Thiran, "Generalization comparison of deep neural networks via output sensitivity," in *ICPR*, 2020, pp. 7411–7418.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.

[38] Z. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *IJCAI*, 2017, pp. 3553–3559.

[39] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *CoRR*, vol. abs/1811.03378, 2018.

[40] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *CoRR*, vol. abs/1810.00069, 2018.

[41] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *ICLR*, 2017.

[42] ——, "Adversarial machine learning at scale," in *ICLR*, 2017.

[43] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[44] X. Mu, K. M. Ting, and Z. Zhou, "Classification under streaming emerging new classes: A solution using completely-random trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1605–1618, 2017.

[45] K. Zhang, C. Gao, L. Guo, M. Sun, X. Yuan, T. X. Han, Z. Zhao, and B. Li, "Age group and gender estimation in the wild with deep ror architecture," *IEEE Access*, vol. 5, pp. 22 492–22 503, 2017.

[46] P. Smith and C. Chen, "Transfer learning with deep cnns for gender recognition and age estimation," in *IEEE BigData*, 2018, pp. 2564–2571.

[47] E. Eidinger, R. Enbar, and T. Hassner, "Age and gender estimation of unfiltered faces," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2170–2179, 2014.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[49] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015, pp. 41.1–41.12.

[50] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[51] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-

Julien, "A closer look at memorization in deep networks," in *ICML*, 2017, pp. 233–242.

X. Supplementary Materials

A. Experiment Setup

Data Sets. We use four datasets to compare the performance of all methods:

(1) **MNIST**[4]: The MNIST is an image dataset of handwritten digits. It contians 10 classes and 784 features;

(2) **20 Newsgroups**[5]: The dataset collates approximately 20,000 newsgroup documents partitioned across 20 different newsgroups. The Word2vec is used to prepossess text data.

(3) **CIFAR-10**[6]: It is a standard classification dataset consisting of 32×32 color images belonging to 10 different object classes.

(4) **VGGFace**[7]: The dataset consists of the crawled images of celebrities on the Web. There are 2622 celebrities in the dataset.

Evaluation Metrics. We use **F-measure** to measure the performance. This measure produces a combined effect of precision (P) and recall (R) of the auditing performance,

$$F\text{-}measure = \frac{2 * P * R}{P + R}.$$

F-measure = 1 if the method identifies all training data with no false positives.

We also employ **AUC**[8] ("Area under the ROC Curve") to access performance. An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots True Positive Rate (TPR) and False Positive Rate (FPR).

B. Results on synthetic data

1) SVM-based model: SVM is a discriminative classifier which classifies new data points by calculating an optimal separating hyperplane[9]. In two dimensional space this hyperplane is a line dividing a plane into two parts each defining a class for data points within it. In this paper, we illustrate with a least squares SVM classifier[10]. Given a set of instance-label pairs (x_i, y_i), $i = 1, \cdots, l$, $x_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$, it solves the following optimization problem: $\min_w \ \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi^2$, subject to the equality constraints: $y_i(wx_i + b) = 1 - \xi_i$, where $C > 0$ is a penalty parameter. We set the partial derivatives of x on the cost:

$$\frac{\partial J(w)}{\partial x} = 2[1 - wx)]w.$$

Then, we can calculate η by Eqn.(5).

2) Tree-based model: We discuss random forest model with completely-random trees as target model in this part[11]. random forest model is usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. Each tree in the classifications takes input from samples in the initial dataset. Features are then randomly selected, which are used in growing the tree at each node. Every tree in the forest should not be pruned until the end of the exercise when the prediction is reached decisively.

Note that because the random forest is a non-linear form, it is not easy to use the gradient to calculate η. In this paper, we attempt to set Gaussian noise perturbation as η to observe this result. Figure 9 shows the complete illustrations.

3) Neural network-baesd: A neural network (NN) is a technique that uses a hierarchical composition of n parametric functions to model an input x. Each function f_i for $i \in 1, \cdots, n$ is modeled using a layer of neurons, which are elementary computing units applying an activation function to the previous layer's weighted representation of the input to generate a new representation. Each layer is parameterized by a weight vector θ_i impacting each neuron's activation. Such weights hold the knowledge of a NN model and are evaluated during its training phase, as detailed below. Thus, a NN defines and computes:

$$M(x) = f_n(\theta_n, f_{n-1}(\theta_{n-1}, \cdots f_2(\theta_2, f_1(\theta_1, x)))) \quad (11)$$

At each layer: $y_j = f(\sum_{i=1}^{3} w_{ij}x_i + b)$, where W_{ij}, x_i and y_j are the weights, input and output respectively. We show a two-linear-layer NN example as follow:

$$\begin{aligned} M(x) &= w_2(w_1 x + b_1) + b_2 \\ &= w_2 w_1 x + w_2 b_1 + b_2 \end{aligned} \quad (12)$$

The above equation can be view as an SVM-based model. We, therefore, apply the same way to calculate η. Figure 10 shows NN-based model results.

Figure 11 shows the results of multiplicative implementation on synthetic data data, the experiment setup is the same as Section VII-B. (a)-(c) are results under black-box condition.

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

[5]http://qwone.com/ jason/20Newsgroups/

[6]https://www.cs.toronto.edu/ kriz/cifar.html

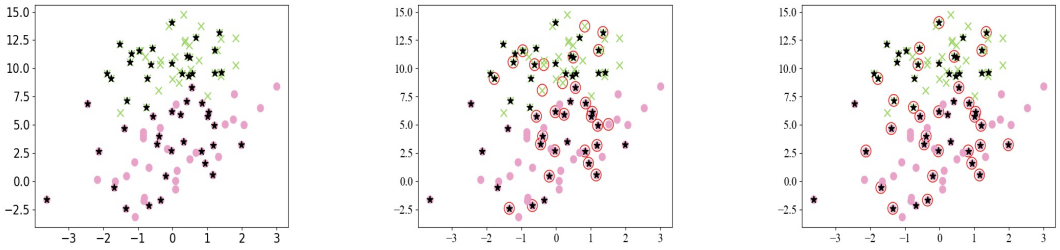
[7]https://www.robots.ox.ac.uk/ vgg/data/vgg_face/

[8]https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[9]Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. Machine Learning 20, 3 (1995), 273–297.

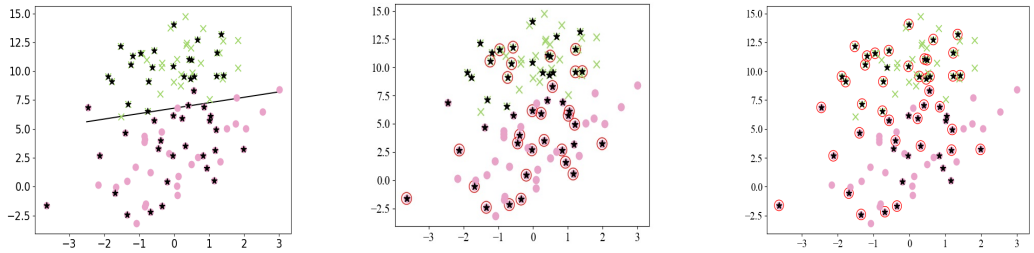
[10]Johan A. K. Suykens and Joos Vandewalle. 1999. Least Squares Support VectorMachine Classifiers.Neural Processing Letters 9, 3 (1999), 293–300.

[11]Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, et. al. 2008. Top 10 algorithms in data mining. Knowledge and Information Systems 14, 1 (2008), 1–37.



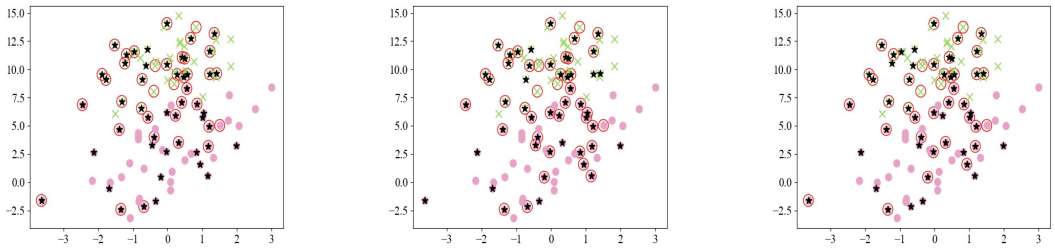
(a) Auditing data. (b) Results on white-box condition. (c) Results on black-box condition.

Fig. 9. An illustration on the Tree-based model. Training data are marked by black star marks training data. Red circle is the auditing results. (b) is result under white-box condition, (c) is result under black-box condition.



(a) Auditing data and hyperplane. (b) Results on white-box condition. (c) Results on black-box condition.

Fig. 10. An illustration on the NN-based model. Training data are marked by black star marks training data. Red circle is the auditing results. (b) is result under white-box condition, (c) is result on black-box condition.



(a) Results on SVM-based model. (b) Results on Tree-based model. (c) Results on NN-based model.

Fig. 11. An illustration on the results of multiplicative implementation. Black star marks training data. red circle marks auditing results. (a) is SVM-based model result, (b) is tree-based model result, (c) is NN-based model result.