Data Provenance via Differential Auditing

Xin Mu, Ming Pang, and Feida Zhu

Abstract—With the rising awareness of data assets, data governance, which is to understand where data comes from, how it is collected, and how it is used, has been assuming ever-growing importance. One critical component of data governance gaining increasing attention is auditing machine learning models to determine if specific data has been used for training. Existing auditing techniques, like shadow auditing methods, have shown feasibility under specific conditions such as having access to label information and knowledge of training protocols. However, these conditions are often not met in most real-world applications. In this paper, we introduce a practical framework for auditing data provenance based on a differential mechanism, 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. Our framework is data-dependent and does not require distinguishing training data from non-training data or training additional shadow models with labeled output data. Furthermore, our framework extends beyond point-based data auditing to group-based data auditing, aligning with the needs of real-world applications. Our theoretical analysis of the differential mechanism and the experimental results on real-world data sets verify the proposal’s effectiveness.

Index Terms—Auditing data, data provenance, machine learning.

I. INTRODUCTION

In an era of accelerated digital transformation, data has been widely recognized as an emerging asset class. In particular, with the rise and sweeping adoption of large models such as GPT-4 [1] and DALL-E [2], the competition in business intelligence has shifted from the design of the ML model to the curation of proprietary data. As a result, understanding where data comes from, how it is collected, and how it can be best used, etc., known as data governance [3], [4], has been assuming ever-increasing importance. It provides insight into the sources and history of data, enabling better decision-making and ensuring responsible use of data assets.

Within the domain of data governance, there is a critical component that has attracted considerable attention from both academia and industry: Auditing Data Provenance (ADP), i.e., how to audit if a given piece of data (which will be referred to as auditing data for the rest of this paper) has been used for training a target machine learning model [5], [6], [7]. With the booming of IoT, 5G, and large language models, data intelligence in the future is bound to be collaborative in nature—models could potentially be trained with data from various sources and domains in data marketplaces or other coordinated settings, e.g., federated learning, and generate real revenue that needs to be attributed to all data contributors. The motivations for ADP, while vary by applications, are therefore manifested most prominently by two critical aspects—(I) Privacy and (II) Incentive governance. 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 contribute and trade data to collectively train models in a collaborative manner, e.g., federated learning in a decentralized variant in which no central nodes are commanding incentivization [8]. The incentive allocation in such a scenario would necessarily entail auditing the usage of all parties’ data to a sufficiently fine granularity to achieve 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 the performance of the model [9]. Unfortunately, most real auditing settings allow access only 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 [10] or the prediction confidence (e.g., class probability) [11]. 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 [5], [6]. 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 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 Fig. 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 [12]. 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 mechanism. 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. DPDA can be viewed as a data dependent method, which needs neither to heavily depend on target model structure nor add an extra data set. This makes it easier to operate on different structures of ML models.

- 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. Meanwhile, a theoretical analysis of differential mechanism on the group-based ADP problem also verifies the effectiveness of the proposal.

- 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 [13], which is to determine if a data record is in the model’s training data set [5], [14]. MIAs has been extensively studied in many fields, e.g., computer vision [10], [11], [15], NLP [6],
and recommender system [17]. While MIAs aims to identify training data from attackers’ perspective, ADP is motivated differently by applications in data-asset-based digital economy [18]. Furthermore, MIAs and ADP take different perspectives to examine the relationship between input data and training model, resulting in different challenges in the technical solution. For example, in data auditing we focus on group-based auditing (versus inferring membership of individual data point) as auditing cares about data granularity at group level rather than individual data point level. By aggregating signals over multiple points, we also improve the result robustness, as illustrated in [19].

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 [7], authors systematically study the impact of 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) [20]. 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 [6], 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 and hence the severe performance degradation of the auditing mechanisms. More recent work [21] analysed the feasibility of membership inference when the model is overfitted or well-generalized and reported a study that discovered overfitting to be sufficient but not a necessary condition for data auditing to succeed.

A closely related research line focuses on utilizing predicted labels from the target model [15], [22]. These methods build on the intuition that training data points exhibit greater robustness compared to non-training data, resulting in smaller prediction variations when training data is perturbed in comparison to non-training data. Compared with our differential mechanism, these methods offer different perspectives and assumptions for tackling the challenges of ADP. For example, Choquette-Choo et al. [15] introduced a label-only attack that instead evaluates the robustness of the model’s predicted (hard) labels under perturbations of the input, to infer membership. In [22], a boundary-attack technique is proposed, which assumes that altering the target model’s predicted labels for member samples incurs a higher cost than for non-member samples. In addition, the heuristic perturbation methods are also adopted to perturb auditing data, like random noise perturbations [15]. In contrast, one of our model’s main contributions is exactly on perturbation design, so that, instead of using random perturbation, we can design specific auditing functions to ensure that our perturbations would result in more obvious differences between training and non-training data, as detailed in Section V.

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 [23], 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 [24]. A recent work [25] showed that adversaries can extract training text from the output of text generation models, indicating memorization threats to user privacy. Though this research area does not directly handle the auditing problem, some of the solutions can be treated as a direct strategy to address this problem, i.e., an auditor has knowledge about model parameters or auxiliary data which is highly related to the model’s training process.

Differential Privacy (DP) [26] 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 [27]. 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 [28], [29]. Some mechanisms add noise to the target function and use the minimum/maximum of the noise function as the output model [30]. It also has been applied to various ML models including tree-based model [31], neural networks [32], [33], and federated learning [34], [35]. 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 $A_e$. 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 if they are both from $A_e$, 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 formulate ADP as follows:

**Definition 3.1. Auditing Data Provenance (ADP):** Given (I) a set of data distributions $A = \{A_1, A_2, \ldots, A_m\}$, (II) a set of groups of data instances $D = \{D_1, D_2, \ldots, D_n\}$, such that for each group $D_i \in D$, $D_i \leftarrow A_k$ for some $1 \leq k \leq m$ and $D_i = \{(x_j, y_j)\}_{j=1}^{p_i}$, 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 $D^T = \{D_{k_1}, D_{k_2}, \ldots, D_{k_l}\}$, $D^T \subset D$, and its correspondent class probability $P_j = \{p_{j1}, p_{j2}, \ldots, p_{jk}\}$ 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
generate a larger difference value (represented 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. [36], [37]. 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.

B. Theoretical Analysis

There are some studies on quantifying the change of the model output by adding a perturbation, e.g., sensitivity analysis [38], [39], [40].

**Lemma 4.1:** [38] Consider a Gaussian perturbation $\Delta x \sim N(0, \varepsilon I)$, the Frobenius norm of the class probabilities Jacobian $||J(x)||_F = \partial M/\partial x^T$, we adopt the Frobenius norm $|| \cdot ||_F$ estimates the average case sensitivity $M$ around $x$:

$$ E_{\Delta x} [||M(x) - M(x + \Delta x)||_F^2 \approx E_{\Delta x} [||J(x)\Delta x||_F^2] = \varepsilon ||J(x)||_F^2. $$

**Lemma 4.2:** [40] 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:

$$ M(x + \Delta x) - M(x) \cong \Delta x \cdot \nabla^T M(x). $$

Shu et al. [39] 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.

In this paper, we provide a theoretical analysis on group-based differential mechanism. We quantify the difference of the differential mechanism in the expected response of a model to any point in the training and non-training data sets. Similar to previous work [19], we first consider a binary classification problem. For any instance-label pair $(x, y)$, we can describe them as:

$$ x = (x_1, x_2) \in \mathbb{R}^{K+L}, \quad x_1 = y \cdot u \in \mathbb{R}^K, $$

$$ x_2 \sim \mathcal{N}(0, \sigma^2 I) \in \mathbb{R}^L, \quad y \sim \{-1, 1\}, $$

where $u \in \mathbb{R}^K$ is a fixed vector. We assume that the first part $x_1$ is sufficient to separate instance from classes $\{-1, +1\}$, and $x_2$ represents the Gaussian noise with variance $\sigma^2$ which has no correlation to the correct label [41]. A linear network $f$ such that for any input $x$, can be described as:

$$ f(x) = w_1 \cdot x_1 + w_2 \cdot x_2. $$

Figure 2. Illustration of differential mechanism.
Note that \( f \) is learned from training data set \( D^T, D^T \sim D \).

**Theorem 4.3:** Given a linear classifier \( f \) trained to classify inputs \((x, y) \in D^T\) (training set), the difference in the expected differential mechanism for data in \( D^T \) and \( D \) is given by
\[
E_{(x,y) \sim D^T}(\kappa) - E_{(x,y) \sim D}(\kappa) = L(\sigma_{\Delta 2})^2, \tag{3}
\]
where \( \kappa = f(x + \Delta x) - f(x) \). \( \Delta x \) is a Gaussian noise with same structure with \( x \), i.e., \( \Delta x = (\Delta_1, \Delta_2) \in \mathbb{R}^{K + L}, \Delta_1 = y \cdot u_\Delta \in \mathbb{R}^K, \Delta_2 \sim \mathcal{N}(0, (\sigma_{\Delta 2})^2 I) \in \mathbb{R}^L \).

**C. The Proof on Theorem 4.3**

Before proofing on theorem 4.3, we describe some preliminaries as follows. Similar to previous work [19], we first consider a binary classification problem. Any instance-label pair \((x, y)\) can be described as:
\[
x = (x_1, x_2) \in \mathbb{R}^{K + L}, \quad x_1 = y \cdot u \in \mathbb{R}^K, \\
x_2 \sim \mathcal{N}(0, \sigma^2 I) \in \mathbb{R}^L, \quad y \sim \{-1, 1\},
\]
where \( u \in \mathbb{R}^K \) is a fixed vector. The \( x_2 \) represents the Gaussian noise with variance \( \sigma^2 \) which has no correlation to the correct label. However, the first part \( x_1 \) is sufficient to separate instance from classes \{−1, +1\} [41].

Here, we discuss the case of a linear ML model in this analysis, i.e., a linear classifier \( f \), such that for any input \( x : f(x) = w_1 \cdot x_1 + w_2 \cdot x_2 \), is learned from training data set \( D^T = \{x_i, y_i | i = 1, 2, \ldots, m\}, \ D^T \sim D \). We assume that the learning algorithm initializes the weights of the classifier \( f \) to zero and the learning rate \( \alpha \) to 1. The training data set is \( D^T = \{x_i, y_i \}_{i=1}^m, D^T \sim D \). The loss is set by \( \mathcal{L}(x_i, y_i) = y \cdot f(x) \). The process of updating weight \( w \) by gradient descent is described by
\[
w_i \leftarrow w_i + \alpha y^{i} x_1^i, \ w_2 \leftarrow w_2 + \alpha y^{i} x_2^i \tag{4}
\]
Based on the above optimization, the learned weights for the classifier \( f \) are given by \( w_1 = \sum_i y^{i} x_1^i = m u \) and \( w_2 = \sum_i y^{i} x_2^i \) irrespective of the training batch size.

For any data point \((x, y)\), we calculate a margin through its label times the classifier’s output \( y \cdot f(x) \). For the data \( x = (x_1, x_2) \sim D \), this margin is as follows,
\[
y \cdot f(x) = y \cdot (w_1 \cdot x_1 + w_2 \cdot x_2) = y \cdot (m u) \cdot (y u) + y \cdot \left( \sum_i y^{i} x_2^i \right) \cdot x_2
\]
\[
= c + y \cdot \left( \sum_i y^{i} x_1^i \right) \cdot x_2 \tag{5}
\]

Let \( \Delta x \) be a Gaussian noise. It has same structure with \( x \), i.e., \( \Delta x = (\Delta_1, \Delta_2) \in \mathbb{R}^{K + L}, \Delta_1 = y \cdot u_\Delta \in \mathbb{R}^K, \Delta_2 \sim \mathcal{N}(0, (\sigma_{\Delta 2})^2 I) \in \mathbb{R}^L \). Then, the margin of \( x + \Delta x \) is as follows,
\[
y \cdot f(x + \Delta x) = y \cdot (w_1 \cdot (x_1 + \Delta_1) + w_2 \cdot (x_2 + \Delta_2)) = y \cdot f(x) + y \cdot f(\Delta x)
\]
\[
= y \cdot (m u) \cdot (y u) + y \cdot \left( \sum_i y^{i} x_2^i \right) \cdot x_2 + y \cdot \left( \sum_i y^{i} x_2^i \right) \cdot \Delta_2
\]
\[
+ y \cdot (m u_\Delta) \cdot (y u_\Delta) + y \cdot \left( \sum_i y^{i} x_2^i \right) \cdot \Delta_2
\]
\[
= c + d + \left( y \cdot \sum_i y^{i} x_2^i \cdot x_2 \right) + \left( y \cdot \sum_i y^{i} x_2^i \cdot \Delta_2 \right) \tag{6}
\]

Next, consider any point in the training set \((x, y) \sim D^T^+ = (x^{(j)}, 1)\) for some index \( j \), we calculated the expected value of the margin for data randomly sampled from the training set. Then, we have:
\[
E_{x(i) \sim D^T} + f(x^{(j)}) = y \cdot c + \mathbb{E}_{x^{(j)} \sim \mathcal{N}(0, \sigma^2)} \left[ \left( \sum_i y^{(i)} x_2^{(i)} \right) \cdot x_2 \right]
\]
\[
+ \mathbb{E}_{x^{(j)} \sim \mathcal{N}(0, \sigma^2)} \left[ y^{(i)} \left( x_2^{(j)} \right)^2 \right] = c + 0 + L \sigma^2 \tag{7}
\]

The details can be found in [19]. Similarly, we calculate the \( x + \Delta x \):
\[
E_{x(i) \sim D^T}^+ + f(x^{(j)} + \Delta x) = c + d + L \sigma^2 + L(\sigma_{\Delta 2})^2 \tag{8}
\]

For data point \((x, 1)\) from \( D^+ \), we have
\[
E_{x(i) \sim D^+} + f(x) = y \cdot c + \mathbb{E}_{x^{(j)} \sim \mathcal{N}(0, \sigma^2)} \left[ \left( \sum_i y^{(i)} x_2^{(i)} \right) \cdot x_2 \right] = c \tag{9}
\]
and
\[
E_{x(i) \sim D^+} + f(x + \Delta x) = y \cdot c + y \cdot d + \mathbb{E}_{x^{(j)} \sim \mathcal{N}(0, \sigma^2)} \left[ \left( \sum_i y^{(i)} x_2^{(i)} \right) \cdot x_2 \right] + \mathbb{E}_{x^{(j)} \sim \mathcal{N}(0, \sigma_{\Delta 2})^2} \left[ \left( \sum_i y^{(i)} \Delta_2^{(i)} \cdot \Delta_2 \right) \right] = c + d \tag{10}
\]
we can hence show that
\[
E_{(x,y) \sim D^T^+}(\kappa) - E_{(x,y) \sim D^+}(\kappa) = L(\sigma_{\Delta 2})^2, \tag{11}
\]
where \( \kappa = f(x + \Delta x) - f(x) \).

Similarly, data points in the \( D^T^- \) and \( D^- \) can been proved. This concludes the proof for Theorem 4.3. \( \square \)

Note that, in this paper, we provide the analysis on the expected margin for data points included in training or not based on a preliminary case. In fact, the analysis can be strengthened from multiple angles. For example, prior works have also argued how over-parametrized deep learning networks memorize training points [24, 42], and some experimental results on the neural network models are discussed in the following and in Section VI-D. The exploration of theoretical analysis on more complex models will be considered as future work.
D. The Observation of Differential Mechanism

To further illustrate the differential mechanism on the complex models, we examine two popular deep structures, GoogLeNet [43] and AlexNet [44] on FashionMNIST and CIFAR-10, to observe the effectiveness of our proposed differential mechanism. First, all auditing data (both training data and non-training data) are processed by adding Gaussian noise. Second, 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 Fig. 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. It is interesting that exceptions like the instance of “frog” in (c) display a contradictory outcome. The experiments involve the introduction of Gaussian noise as the perturbation, which introduces a randomized factor into the process. In scenarios where the noise introduced is very slight, its impact on training data may not be significant. This scenario could lead to smaller difference values for training data, potentially even resulting in values of difference that are lower than those observed for non-training data. This observation also strongly emphasizes the critical need for a thoughtfully crafted perturbation for auditing data.

V. THE PROPOSED FRAMEWORK 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 Fig. 1, DPDA comprises two main steps: (I) Auditing data is first processed by the auditing function; (II) The differential value is calculated for the target model’s output between original data and processed data. Finally, the differential values are 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 5.1. 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 \( \forall x, x' \in \mathbb{R}^d, A(x) = A(x') \Rightarrow x = x' \).

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 designs 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 [45] or activation function [46].

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.

After the auditing function is confirmed, we define \( \Phi() \) as a differential calculation function such that \( \Phi(M(A(D_i)), M(D_i)) \) is the difference of target model’s output between original data group and processed data group. If the input of \( \Phi \) is group-based data, the output will describe group-based difference, e.g., the mean of the differential value of each data point in a group. In this paper, the model output takes the maximum probability within it.

In the following, we explore two implementations under this framework: one additive and one multiplicative.

A. Additive Implementation

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

\[
A(x) = x + \varepsilon \eta
\]
Algorithm 1: Additive Implementation.

| Input | \( D = \{D_1, D_2, \ldots, D_e\} \) - auditing data, \( M \) - target model |
|-------|-------------------------------------------------------------------|
| Output | \( D_i \) - training data |
| 1 | if \( M \) is not available then |
| 2 | \( O \leftarrow M(D) \), // Calculate the output. |
| 3 | \( M_v \leftarrow \) train a model on \( \{D, O\} \) |
| 4 | for \( i = 1, \ldots, e \) do |
| 5 | \( D'_i \leftarrow A(D_i) \), // using Eqn.\((12)\). |
| 6 | \( S_i \leftarrow \Phi(M(D'_i), M(D_i)) \) |
| 7 | end |
| 8 | if \( \forall_j, S_j > \text{threshold} \) then \( D_j \leftarrow \) training data; |
| 9 | else \( D_j \leftarrow \) non-training data |

where \( \varepsilon \) is a slack variable and \( \eta \) is an offset. Note that the purpose of introducing \( \eta \) 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 [47]. 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 [48], [49]. Goodfellow et al. [50] provided a strategy to use the linear view to generate adversarial inputs. Let \( \theta \) be the parameters of the model, \( x \) be the input to the model, \( y \) be the label associated with \( x \), \( J(\theta) \) be the cost to train the model, \( \varepsilon \) is a slack variable and the adversarial example can be defined by

\[
x' = x + \varepsilon \cdot \text{sign}(\nabla_x J(\theta))
\]

Following this idea, we set \( \eta \) in (12) as the sign of the target model’s cost function gradient \( \eta = \text{sign}(\nabla_x J(\theta)) \). This setting would maximize the loss function and result in the greatest misclassification for the auditing data processed by \( A(\cdot) \).

Algorithm 1 illustrates the sketch of the additive implementation. Lines 5–10 show the case when the target model is under the white-box assumption. In line 7, \( S_i \) is equal to the mean of differential value in each group data. 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 \( \eta \). The process is described in Algorithm 1 Lines 1–4.

Algorithm 2: Determining Threshold.

| Input | \( Q \) - a list with \( e \) elements, \( \hat{\tau} \) - initialize to a larger value. \( \sigma(\cdot) \) - standard deviations |
|-------|-------------------------------------------------------------------|
| Output | \( t \) - threshold |
| 1 | for \( i = 1, \ldots, e \) do |
| 2 | \( Q_i \leftarrow Q[1:i], Q_r \leftarrow Q[i:e] \) |
| 3 | \( \tau \leftarrow |\sigma(Q[1:i]) - \sigma(Q[i:e])| \) |
| 4 | if \( \tau < \hat{\tau} \) then |
| 5 | \( t \leftarrow Q[i] \) |
| 6 | \( \hat{\tau} \leftarrow \tau \) |
| 7 | end |
| 8 | end |

Threshold determination: In Algorithm 1, it is crucial to highlight that Line 9 necessitates the inclusion of a threshold. This threshold is pivotal for determining the inclusion of auditing data within the training data category. As emphasized in our earlier discussions, our anticipation is that training data will exhibit greater differential values compared to non-training data. In essence, this expectation implies that \( \{S_1, S_2, \ldots, S_e\} \) should distinctly form two distinct groups within the distributions. We employ the method described in [51] to determine the optimal threshold for effectively distinguishing the two groups. The procedure we adopt is as follows: (1) We initiate by generating a list \( Q \) comprising all the values in \( \{S_1, S_2, \ldots, S_e\} \), arranged in descending order; (2) A threshold \( \tau \) selected from this list subsequently results in the formation of two sub-lists, denoted as \( Q^l \) (the left sub-list) and \( Q^r \) (the right sub-list); (3) The primary criterion utilized in this context is the minimization of the difference in standard deviations \( \sigma(\cdot) \):

\[
\hat{\tau} = \text{arg min}_\tau |\sigma(Q^r) - \sigma(Q^l)|
\]

The threshold \( \hat{\tau} \) is crucial for distinguishing between training and non-training data. Specifically, larger values align with the training data category, while smaller values pertain to non-training data. The procedural intricacies regarding the determination of this threshold can be found in Algorithm 2, providing a comprehensive guide to its calculation. To further elucidate its application, a practical example is furnished in Section VI-B. Note that our focus is on auditing data that includes both training and non-training data. For other audit scenarios, we provide a discussion in Section VII.

B. Multiplicative Implementation

Alternatively, the ADP problem can be treated as a ranking problem by solving the following objective:

\[
\max_g \quad g(D_i, D_j)
\]

s.t. \( D_i \in_A D^T, D_j \notin_A D^T \)

\[
D_i, D_j \in D, 1 \leq i, j \leq n
\]

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...
Algorithm 3: Multiplicative Implementation.

Input: \( \{D_1, D_2, \ldots, D_x\} \) - auditing data, \( M \) - target model, \( D_t \) - initialization of training data.

Output: \( D_t \) - training data

1 while Maximum number of iterations do
2 \( S_t \leftarrow \{\Phi(\langle M(W_{D_t}), M(D_t)\rangle)\}_{t=1}^{\infty} \)
3 if \( \forall_j, S_j > \text{threshold} \) then \( D_j \leftarrow \) training data;
4 else \( D_j \leftarrow \) non-training data;
6 end

...a similarity function to maximize the difference between data belonging to the target model \( M' \)'s training and non-training data.

In the multiplicative implementation, we can search for auditing function \( A() \) by the following objective function:

\[
\max_A \Phi(\langle M(A(D_t)), M(D_t)\rangle) - \Phi(\langle M(A(D_o)), M(D_o)\rangle)
\]  

(15)

where \( M() \) is the target model output, \( A() \) is the auditing function, \( D_t \) is \( M' \)'s training data, \( D_o \) is the non-training data. Since (15) 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:

\[
A(x) = Wx,
\]  

(16)

where \( W \in \mathbb{R}^{d \times d} \).

Optimization: It is important to note that (15) 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 \( \Phi \) of auditing data \( D \). Then calculate a threshold (as mentioned in the section V-A) to separate the value \( \Phi \) 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 (15) as follows:

\[
W' = W - \epsilon \frac{\partial L(W)}{\partial W}
\]  

(17)

where

\[
\frac{\partial L(W)}{\partial W} = [M(W_{D_t}) - M(D_t)]M'(W_{D_t})D_t
\]  

\[ - [M(W_{D_o}) - M(D_o)]M'(W_{D_o})D_o \]  

(18)

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

3) Calculate the threshold (Algorithm 2) to separate the value \( \Phi \) 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 sketch of the process is described in Algorithm 3. Note that when \( M' \) is not available under the black box assumption, a simulation model can be employed.

VI. EXPERIMENT

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 contains 10 classes and 784 features.

2) 20 Newsgroups\(^2\): The dataset collates approximately 20,000 newsgroup documents partitioned across 20 different newsgroups. The Word2vec is used to preprocess text data.

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

4) VGGFace\(^4\): The dataset consists of the crawled images of celebrities on the Web. There are 2622 celebrities in the dataset.

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 prediction class probability). The criterion is like a preset threshold, the data with higher probability is treated as training data.

2) Shadow Learning Technique (SLT) \(^5\): 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) Boundary-Attack technique (BA) \(^\[22\]\): BA considers that the cost of changing the target model’s predicted labels for member samples is larger than the cost for non-member samples. It can simply decide on member samples based on the magnitude of the perturbations.

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

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

6) 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. The threshold is determined by Algorithm 2, i.e., all prediction values of auditing data can be regarded as the list \( Q \), and it is used in Algorithm 2.

---

\(^1\)http://yann.lecun.com/exdb/mnist/

\(^2\)http://qwone.com/jason/20Newsgroups/

\(^3\)http://www.cs.toronto.edu/~kriz/cifar.html

\(^4\)http://www.robots.ox.ac.UK/~vgg/data/vgg_face/
In SLT, the codes are developed based on the original paper.\(^3\) We employ 50 shadow models and one attack model which is SVM\(^6\) with RBF kernel and other parameters set by default values. In BA, we use Hop\&Skip\&Jump\(^7\) to find perturbations to change their predicted labels and a threshold-choosing method mentioned in the paper to make final auditing results. In DPDA-RN, random values are set by Gaussian noise. The parameter \(\varepsilon\) 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 \(\varepsilon\). In DPDA-MUL, \(W\) is initialized to a semi-positive definite matrix.

**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, \(F\)-measure = \(\frac{2\cdot P \cdot R}{P+R}\). F-measure = 1 if the method identifies all training data with no false positives. We also employ AUC (“Area under the ROC Curve”) to access performance. A 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).

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: Let \(N\) be the total number of groups of auditing data; \(A_i\) be the result of a group, which is calculated by the average results of each group. Then, we can recall the python tool\(^8\) to calculate the AUC or F-measure results, e.g., \(F\)-measure\((\{A_i\}_{i=1}^N, \{A_i^*\}_{i=1}^N), AUC(\{A_i\}_{i=1}^N, \{A_i^*\}_{i=1}^N)\), where \(A_i^*\) is truth label.

### B. Results on Synthetic Data

We take SVM-based, Tree-based and NN-based models as the target model and train them on a two-dimensional synthetic dataset with 100 data points. Fig. 4 shows SVM-based model results and others are provided in Appendix A, available online. We indicate two kinds of data in green and red respectively, and mark training data by “black star”. The auditing data are in the form of groups auditing data by 10 points in a group. The auditing problem is to identify the “black star” training data given all data points as the input auditing data.

SVM is a discriminative classifier which classifies new data points by calculating an optimal separating hyperplane \([52]\). 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 the least squares SVM classifier \([53]\). Given a set of instance-label pairs \((x_i, y_i), i = 1, \ldots, l, x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}\), it solves the following optimization problem: \(\min_w \frac{1}{2}w^TW + C \sum_{i=1}^l \xi_i\), subject to the equality constraints: \(y_i(x_i^Tw + b) = 1 - \xi_i\), where \(C > 0\) is a penalty parameter. We set the partial derivatives of \(w\) on the cost: \(\frac{\partial J(w)}{\partial x} = 2[1 - w^Tx]\). Then, we can calculate \(\eta\) by \(\eta = \text{sign}\left(\nabla_x J(\theta)\right)\).

Fig. 4(a) shows the target model by a solid black line and a simulation model by blue dotted lines. The effectiveness of the proposed method is demonstrated by the auditing results as labeled with red circles in Fig. 4, in which (b) shows the WB condition and (c) shows the BB condition. In addition, we show the threshold setting in Fig. 4(d). The 100 points in the synthetic data set are divided into training and non-training groups with the ratio of 1:1. Fig. 4(d) shows an example of the distribution for \(\tau = (\sigma(Q^\prime) - \sigma(Q))\) 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 1:1. 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 1 provides more comprehensive results on SVM-based, Tree-based and NN-based target model. It is evident that the DPDA consistently outperforms all other methods.

---

\(^3\)https://github.com/spring-epfl/mia
\(^4\)https://scikit-learn.org/stable/modules/svm.html
\(^5\)https://github.com/Trusted-IA/adversarial-robustness-toolbox
\(^6\)https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics
TABLE I
RESULTS OF DIFFERENT TARGET MODELS ON DIFFERENT DATA SETS

| Algorithm | MNIST | 20 Newsgroups | CIFAR10 |
|-----------|-------|---------------|---------|
| 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 |
| BA        | 0.592 ± 0.03 | 0.725 ± 0.05 | 0.594 ± 0.02 | 0.641 ± 0.03 | 0.654 ± 0.04 | 0.682 ± 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 |

SVM-based model

| Algorithm | MNIST | 20 Newsgroups | CIFAR10 |
|-----------|-------|---------------|---------|
| 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 | 0.652 ± 0.06 | 0.652 ± 0.06 | 0.651 ± 0.07 | 0.673 ± 0.04 |
| BA        | 0.674 ± 0.03 | 0.654 ± 0.02 | 0.638 ± 0.03 | 0.660 ± 0.02 | 0.664 ± 0.02 | 0.679 ± 0.07 |
| 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 |

Tree-based model

| Algorithm | MNIST | 20 Newsgroups | CIFAR10 |
|-----------|-------|---------------|---------|
| 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 |
| BA        | 0.602 ± 0.02 | 0.725 ± 0.04 | 0.625 ± 0.02 | 0.648 ± 0.03 | 0.654 ± 0.05 | 0.630 ± 0.03 |
| 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 | 0.752 ± 0.01 | 0.719 ± 0.01 | 0.810 ± 0.04 |

NN-based model

| Algorithm | MNIST | 20 Newsgroups | CIFAR10 |
|-----------|-------|---------------|---------|
| 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 |
| BA        | 0.602 ± 0.02 | 0.725 ± 0.04 | 0.625 ± 0.02 | 0.648 ± 0.03 | 0.654 ± 0.05 | 0.630 ± 0.03 |
| 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 | 0.752 ± 0.01 | 0.719 ± 0.01 | 0.810 ± 0.04 |

ADD and MUL have #wins/#draws/#losses | 3/3/0 | 6/0/0 | 5/1/0 |

in almost all cases. The closest competitor, SLT, which relies on the shadow model technique, falls short of DPDA. DPDA-RN, based on random perturbation, consistently lags behind DPDA across all datasets. 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 depends on model shadow 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.

- BA is an algorithm that continuously seeks perturbations capable of altering the model’s prediction and employs these perturbations for auditing data. The result shows that it falls short in comparison with DPDA. One possible reason for this discrepancy is that BA relies on certain assumptions, such as model robustness or the distribution of data samples and the model’s boundary. These assumptions encounter difficulties in scenarios where model boundaries are less well-defined. Additionally, the process of perturbation search in BA is computationally expensive, making it less practical for real-world applications.

- 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. In the tree-based model experiments conducted on the MNIST dataset, we have observed the performance of these two (SLT and DADP) methods becomes notably similar. One potential explanation for this phenomenon could be the inherent challenges faced by tree-based models in effectively memorizing training data due to their limited performance capabilities.3 In addition, we also see that MUL achieves higher performance, and 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 [54], [55], [56]. 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 [57], including 18 layers deep neural network. And it is trained on VGG-face dataset [58]. 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.

Setting: In the experiment, we used 20 people with 20 k images as training data to train a gender estimation classifier.

3Note that the aspect related to the memorization of the target model, data similarity, and the ADP problem has been discussed in the section VII (b)
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 person are naturally regarded as a group $D_i$. As an example shown in Fig. 5, person 1 and person 2 have been used to train the target model, their images are shown in Fig. 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: Fig. 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 $[100, 200, 500, 1000]$. First, 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. Second, 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 here demonstrate the reasonableness and feasibility of this method on the group-based auditing problem.

### Time Complexity

As we mentioned before, unlike shadow-based techniques that involve learning a combination of multiple shadow models to simulate the target model, our framework derives data transformations directly from the data using a single auditing function. This difference in approach contributes to the potential efficiency of our method. In this section, we have conducted an experiment about the efficiency of the proposed mechanism as follows: We evaluate the time complexity in the training stage. The model configuration for this experiment remains consistent with the setup utilized in the benchmark dataset experiment (Section VI-C).

We now proceed to showcase the results derived from our experiment, focusing on the NN-based model applied to the CIFAR10 dataset. In the training stage, shadow-based methodologies often necessitate the training of multiple shadow models to replicate the target model’s behavior. Conversely, our approach’s computational demands solely revolve around the target model itself, as manifested in (13) or (18). The information presented in Table II underscores that SLT necessitates training multiple shadow models, resulting in a time-consuming training phase when compared to our method.

### Parameter Analysis

F. 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.

Fig. 7(a) describes the number of different sizes of classes in auditing data set and Fig. 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 is 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. Fig. 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 vectors have a relatively small impact on performance.
In conclusion, we describe the relation of data similarity, target model memorization and differential mechanism in Fig. 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.

(3) Audit scenario: In practical settings, the auditing data could form any distribution, some of which even unfriendly for our auditing purpose, e.g., when the auditing data are either all training data or all non-training data. In this paper, particularly in Algorithm 2, our focus is on auditing data that includes both training and non-training data, which by assumption, should form different data groups. For other forms of auditing data, we provide a plausible solution as follows, and will consider this challenging issue for future work.

For example, when all auditing data belongs exclusively to either the training data or non-training data category, this implies that all elements in set \( Q \) (representing difference values) may originate from the same distribution. As a result, our focus shifts to the crucial task of effectively determining whether the elements in \( Q \) indeed follow the same distribution. Fortunately, there exist well-established statistical methods that can assist in addressing this challenge. For instance, techniques like the \( \text{kstest}^{10} \) or \( t\text{-test}^{11} \) can be valuable tools in this context. Before implementing Algorithm 2, we can employ a statistical test \( \text{kstest} \) to determine if two subsets, denoted as \( Q_t \) and \( Q_r \), are drawn from the same distribution. The procedure is as follows: \( p\text{-value} \leftarrow \text{kstest}(Q_t, Q_r) \). Following the iterative process, if the average \( p\text{-value} \) is notably high, this indicates that all auditing data may originate from the same distribution. To further discern whether the distribution corresponds to the training data or the non-training data category, one potential approach could involve using randomly generated data as representatives of non-training data and computing the difference values on these synthetic data. Subsequently, a threshold can be established by selecting the top \( t \) difference values and comparing it with the difference value obtained from the auditing data to make an inference about its category. Note that this solution is very preliminary, it indeed raises an interesting and valuable challenge. We intend to consider this challenging issue in our future work.

VIII. Conclusion

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

---

**TABLE III**

| Data similarity | ‘cat’   | ‘cat’   | ‘truck’ | ‘dog’  | ‘horse’ | ‘plane’ | ‘ship’  | ‘deer’  | ‘bird’  | ‘frog’  |
|----------------|--------|--------|---------|--------|---------|---------|---------|---------|---------|---------|
| Accuracy       | 66.05  | 63.73  | 62.44   | 61.86  | 60.99   | 58.83   | 57.207  | 55.78   | 54.87   | 54.21   |
| \( p\text{-value} \) | ***    | ***    | ***     | ***    | ***     | ***     | ***     | ***     | ***     | 0.507   |

(Data similarity is calculated by the average pair-wise euclidean distance between training and non-training data, i.e., the smaller the value, the higher the similarity. The meaning of \( p\text{-value} \) is the same as Fig. 3.)

---

10https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest. html
11https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind. html
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.

ACKNOWLEDGMENT

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] OpenAI, “GPT-4 technical report,” 2023, arXiv:2303.08774.
[2] A. Ramesh et al., “Zero-shot text-to-image generation,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 8821–8831.
[3] J. Pei, F. Zhu, Z. Cong, X. Luo, H. Liu, and X. Mu, “Data pricing and data asset governance in the AI era,” in Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining, 2021, pp. 4059–4059.
[4] Y. Chandrasekaran, H. Iia, A. Thudi, A. Travers, M. Yaghini, and N. Papernot, “SoK: Machine learning governance,” 2021, arXiv:2109.10870.
[5] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in Proc. IEEE Symp. Security and Privacy, 2017, pp. 3–18.
[6] C. Song and V. Shmatikov, “Auditing data provenance in text-generation models,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2019, pp. 196–206.
[7] M. A. Rahman, T. Rahman, R. Laganière, and N. Mohammed, “Membership inference attack against differentially private deep learning model,” Trans. Data Privacy, vol. 11, no. 1, pp. 61–79, 2018.
[8] Q. Yang, “Federated Learning: Privacy and Incentive,” Berlin, Germany: Springer, 2020.
[9] C. Song, T. Ristenpart, and V. Shmatikov, “Machine learning models that remember too much,” in Proc. ACM SIGKDD Conf. Comput. Commun. Secur., 2017, pp. 587–601.
[10] M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning,” in Proc. IEEE Symp. Secure Privacy, 2019, pp. 739–753.
[11] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, “MI-leaks: Model and data independent membership inference attacks and defenses on machine learning models,” in Proc. Netw. Distrib. Syst. Secur. Symp., 2019.
[12] S. Shejwalkar, H. A. Iana, A. Houmansadr, and R. Sim, “Membership inference attacks against NLP classification models,” in Proc. Int. Conf. Neuro Inf. Process. Syst. Workshop Privacy Mach. Learn., 2021.
[13] H. Hu, Z. Salcic, G. Dobbie, and X. Zhang, “Membership inference attacks on machine learning: A survey,” 2021, arXiv:2103.07853.
[14] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, “Towards demystifying membership inference attacks,” 2018, arXiv:1807.09173.
[15] C. A. Choquette-Choo, F. Tramèr, N. Carlini, and N. Papernot, “Label-only membership inference attacks,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 1964–1974.
[16] X. Pan, M. Zhang, S. Ji, and M. Yang, “Privacy risks of general-purpose language models,” in Proc. IEEE Symp. Secure Privacy, 2020, pp. 1314–1331.
[17] M. Zhang et al., “Membership inference attacks against recommender systems,” in Proc. ACM SIGMCS Conf. Comput. Commun. Secur., 2021, pp. 864–879.
[18] J. Pei, “Data pricing - from economics to data science,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 3553–3554.
[19] P. Maini, M. Yaghini, and N. Papernot, “Dataset inference: Ownership resolution in machine learning,” in Proc. Int. Conf. Learn. Representations, 2021.
[20] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro, “LOGAN: Membership inference attacks against generative models,” in Proc. Privacy Enhancing Technol., 2019, pp. 133–152.
[21] Y. Long et al., “Understanding membership inferences on well-generalized learning models,” in Proc. ACM SIGMCS Conf. Comput. Commun. Secur., 2021, pp. 880–895.
[22] Z. Li and Y. Zhang, “Membership leakage in label-only exposures,” in Proc. ACM SIGMCS Conf. Comput. Commun. Secur., 2021, pp. 880–895.
[53] J. A. K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” Neural Process. Lett., vol. 9, no. 3, pp. 293–300, 1999.
[54] K. Zhang et al., “Age group and gender estimation in the wild with deep Vror architecture,” IEEE Access, vol. 5, pp. 22492–22503, 2017.
[55] P. Smith and C. Chen, “Transfer learning with deep CNNs for gender recognition and age estimation,” in Proc. IEEE Int. Conf. Big Data, 2018, pp. 2564–2571.
[56] E. Eidinger, R. Enbar, and T. Hassner, “Age and gender estimation of unfiltered faces,” IEEE Trans. Inf. Forensics Secur., vol. 9, no. 12, pp. 2170–2179, Dec. 2014.
[57] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[58] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” in Proc. Brit. Mach. Vis. Conf., 2015, pp. 41.1–41.12.
[59] D. Arpit et al., “A closer look at memorization in deep networks,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 233–242.
[60] X. Wu et al., “Top 10 algorithms in data mining,” Knowl. Inf. Syst., vol. 14, no. 1, pp. 1–37, 2008.

**Xin Mu** received the PhD degree in the LAMDA Group from Nanjing University, advised by Zhi-Hua Zhou. He is currently working as a senior researcher with Peng Cheng Laboratory, Shenzhen, China. From 2015 to 2018, he worked as a research assistant in the LARC lab at Singapore Management University, supervised by Prof. Ee-Peng Lim and Prof. Feida Zhu. His primary research focus lies in the fields of machine learning and data mining, with ongoing projects centered around ensemble methods, stream mining, and data governance.

**Ming Pang** received the BSc and PhD degrees in computer science and technology from Nanjing University, Nanjing, China, in 2014 and 2020, respectively. Currently, He is working as a staff engineer with the JD Group. His research interest is mainly on machine learning and data mining. He is currently working on information retrieval and query understanding.

**Feida Zhu** received the PhD degree in computer science from the University of Illinois, Urbana-Champaign (UIUC), in 2009. He is currently a tenured associate professor and associate dean at the School of Computing and Information Systems, Singapore Management University. His research interests include AI and collaborative intelligence, blockchain, data asset and AI governance, with emphasis on their application to business, financial and consumer innovation. He has more than 100 peer-reviewed research publications at top international venues, including ICDE, VLDB, SIGMOD, KDD, WWW, IMLR, TODS, TKDE, etc. His work on large-scale frequent pattern mining has won the Best Student Paper Awards at 2007 IEEE International Conference on Data Engineering (ICDE’07) and 2007 Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD’07). He won the Early Career Award of PAKDD’19 and is the General Co-Chair of IEEE ICDM’18 and ACM KDD’21.