Catch Me if You Can: Detecting Unauthorized Data Use in Deep Learning Models

Zitao Chen
University of British Columbia

Karthik Pattabiraman
University of British Columbia

Abstract

The rise of deep learning (DL) has led to a surging demand for training data, which incentivizes the creators of DL models to trawl through the Internet for training materials. Meanwhile, users often have limited control over whether their data (e.g., facial images) are used to train DL models without their consent, which has engendered pressing concerns.

This work proposes *MembershipTracker*, a practical data provenance tool that can empower ordinary users to take agency in detecting the unauthorized use of their data in training DL models. We view tracing data provenance through the lens of membership inference (MI). *MembershipTracker* consists of a lightweight data marking component to mark the target data with small and targeted changes, which can be strongly memorized by the model trained on them; and a specialized MI-based verification process to audit whether the model exhibits strong memorization on the target samples.

Overall, *MembershipTracker* only requires the users to mark a small fraction of data (0.005%∼0.1% in proportion to the training set), and it enables the users to reliably detect the unauthorized use of their data (average 0% FPR@100% TPR). We show that *MembershipTracker* is highly effective across various settings, including industry-scale training on the full-size ImageNet-1k dataset. We finally evaluate *MembershipTracker* under multiple classes of countermeasures.

1 Introduction

Modern deep learning (DL) models have attained remarkable performance in various tasks such as image classification and language generation. Their success is largely driven by the availability of massive training data [50, 52]. However, it is not always clear whether the data used to train these models has been used with the permission of the data owners. For instance, Clearview.ai, in developing their facial recognition system, scraped billions of user photos from social media platform without the users' consent [26]. The unregulated use of personal data for building DL models has engendered mounting concerns [20, 26, 66], and is in violation of privacy laws such as the European Union's General Data Protection Regulation (GDPR) [2]. In spite of this, data holders often have limited agency in detecting the unauthorized use of their data due to the lack of practical data provenance tools.

This leads to two main lines of work for data provenance. The first line of work seek to inject designated features into the user data, which can induce the model to exhibit a certain detectable behavior for provenance purposes [27, 39, 53, 68]. The injected features can take the form of a backdoor trigger pattern [27, 39]; or a spurious feature pattern that can cause output probability shift by the model [68]. However, these techniques either impose strong assumptions on the users (e.g., the ability to mark a large fraction of data) [27, 39, 53] and/or suffer from limited auditing performance [39, 53, 68].

The second line of work is based on *membership inference* (MI). There are instance- and user-level MI - the former infers whether a *specific* sample is used to train a model [8, 56]; while the latter detects the usage of *any* of the user's data [12, 32, 58]. Our work focuses on auditing whether *specific user samples* are used for training the target model. Unfortunately, existing MI methods often suffer from limited effectiveness, as measured by their true positive rate (TPR) and false positive rate (FPR) [8]. However, a high-power MI method is needed to facilitate data provenance with low FPR (to avoid false accusation) *and* high TPR (for correct data provenance).

To improve, recent studies propose several techniques based on data poisoning [10, 13, 65]. Their common idea is to inject mislabeled samples into the model's training set, which can amplify the model's behavioral changes on some target samples, thereby making them easier to be de-identified. To apply these techniques for data provenance, the users have to be able to mislabel some training data, which, unfortunately, excludes the real-world scenarios where the data labels are assigned by the model owner or an external service [19, 21]. This renders these techniques challenging for real-world use.

This work. We present *MembershipTracker*, a MI-based data provenance tool that can overcome the aforementioned challenges, with a two-step process as illustrated in Fig. 1.

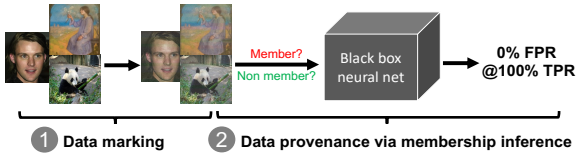


Figure 1: *MembershipTracker* is a data provenance tool that operates by: (1) marking the target data (e.g., facial images, artworks) with small and targeted changes, and then (2) initiating a specialized membership inference process to audit whether the target data are used for training the model.

MembershipTracker operates under the **realistic setting** where we assume the users have neither the capability nor expertise to train shadow/proxy models [6, 8, 70], or to manipulate the data labels [65], as these are well beyond the capacity of ordinary personal users (Section 3.1.1). Rather, *MembershipTracker* only requires the users to mark a small fraction of data (amounting to 0.005%∼0.1% of the training set) for provenance purposes. Our goal is to empower the users to carry out high-power MI on their target data with high TPR and low FPR, given *black-box* access to the model.

Technical design. We first observe that the effectiveness of MI on a given model is directly related to the model's ability in memorizing individual data points, and hence samples that are strongly memorized by the model (e.g., outlier samples) are easier to be de-identified [8, 14, 65, 71]. Therefore, in order to fulfill our goal, there are two challenges as follows.

1. How can the users *amplify* the model's memorization on the target data with minimal data modification?
2. How can the users reliably *audit* whether the model exhibits strong memorization on their target data?

For the first challenge, we propose a lightweight data marking technique to mark the target data with small and targeted changes, which are formulated as the combination of outlier feature and procedural noise [17]. These subtle changes can be readily created by the users without assuming access to a proxy model, and can still induce the model to strongly memorize the specially-marked samples while preserving their visual information (e.g., Fig. 1).

For the second challenge, given a target user's small set of marked samples, we propose a novel *set*-based MI process to audit whether they are used to train the model. Our approach follows a common idea in existing user-level MI studies [32, 47, 58] to leverage the *collective* information across the small set of target samples for MI. But unlike prior methods, *MembershipTracker* does not require any additional shadow/reference models [32, 47, 58] and can enable high-power MI for reliable data provenance.

Contributions. We make three contributions as follows.

- Propose a lightweight data marking technique to mark the users' target data with small and targeted changes, which can be strongly memorized by the model trained on them.
- Develop a high-power set-based MI verification process that can reliably audit whether the target data are used to train the model (with low FPR and high TPR).
- Integrate the above as a tool called *MembershipTracker*, and evaluate its effectiveness across a variety of settings (six benchmark datasets and six DL architectures). We also comprehensively evaluate a total of six classes of countermeasures.

We find that by merely marking a small fraction of samples (0.005%∼0.1% of the dataset), *MembershipTracker* effectively empowers the users to trace the provenance of their data with minimal false positives (average 0% FPR@100% TPR), and it is also scalable to the large-scale ImageNet training (on both Convolutional and Transformer network). This renders *MembershipTracker* a practical data provenance tool and contributes to responsible AI practice.

2 Related Work

Our work focuses on detecting the unauthorized use of personal data in training DL models, which is complementary to existing work that aim to protect the data from unwanted use by rendering them un-learnable [30, 55]. Hence we focus on existing literature for data provenance, and we classify them into two categories.

Non membership-inference based solutions. We divide them into dataset- and user-level solutions.

Dataset level provenance aims to detect whether a DL model is trained on a specific dataset [39, 46, 53]. Maini et al. propose dataset inference, which detects whether two models trained on the same dataset exhibit similarities in their decision boundaries [46]. Other work modify portions of training set to leave a certain artifact on the models, such as radioactive data [53] and backdoor watermark [39]. However, recent work [31] finds that these techniques [39, 53] still have poor auditing performance.

Moreover, dataset level solutions require several assumptions that are outside the scope of our work. First, many of them require control over a nontrivial portion of training set (e.g., 10%∼20% [39, 53]). Some solutions assume access to some known feature extractor [53], access to the training set or the ability to train a proxy model [46, 53]. These are well beyond the user capability we consider (in Section 3.1.1).

User level provenance seeks to detect whether the users' data are used to train a DL model [27, 68], and they require only modifying the users' own data. Their idea is to mark the users' data with a designated feature to induce a detectable behavior into the model. Hu et al. propose to inject backdoor trigger feature into the user data [27], but they assume the user capability to manipulate the data labels. Other studies on clean-label backdoor attacks may be repurposed to overcome this, but they still require training a proxy model to compute the poisoned data [59, 73].

Wenger et al. propose to inject a target spurious feature into the user data to be learned by the suspected model, and then detect whether the model has associated the injected feature with the target class label [68]. To detect, the users can separately overlay the target and non-target features to some auxiliary data (outside the target class), and then determine whether the target feature causes a slightly higher prediction probability on the target class, compared with the non-target features (e.g., 10% vs. 1%). Both their work and ours involve adding spurious features to mark users' data, but there are several major differences between the two. Notably, our analysis reveals that they [68] significantly *underestimate* their detection false positives (from 0% to >30%). This is because they employ a *discrepant* detection procedure for the testing features that are used for training (true positives), and those that are not (false positives); however, a fair evaluation should follow a consistent procedure (details in Appendix D). Moreover, we also realign their technique for MI-based data provenance in our work, and find that even when their approach has incurred higher distortion to the data, it still has poorer MI performance than *MembershipTracker* (Appendix D).

Membership-inference (MI) based solutions. MI also represents a natural fit for tracing data provenance, and there are instance- and user-level MI solutions.

Instance-level MI detects whether a specific instance is used to train a model. It was first proposed by Shokri et al. [56] and there are many follow-up studies [6, 8, 16, 70, 71]. However, even state-of-the-art MI methods [6, 8, 70] still struggle with achieving high TPR when controlled at the low FPR regime (more in Section 4.1).

This has led to several attempts at improving the effectiveness of existing MI methods via data poisoning. Tramer et al. propose to inject mislabeled samples into the model's training set, which can transform the target samples into outliers and amplify their influence on the model's decision, thereby making the target samples easier to be de-identified [65]. While injecting mislabeled samples represents an effective solution for improving the MI success [10, 13, 65], it may be challenging for the users to apply to their data, because in many scenarios the users may not have control over the data labels [19, 21, 68]. Even if this is feasible, we find that such a method can still only increase the MI success to a limited extent (validated in Appendix E). By contrast, *MembershipTracker* does not assume control of the data labels, and is still able to facilitate high-power MI for tracing data provenance.

User-level MI aims to detect whether any of a user's data is used to train a model. Unlike instance-level MI, the auditor possesses a set of samples from a target user that are *not* necessarily used to train the target model. Song et al. [58] propose the first user-level MI for natural language models, which first trains multiple shadow models, and then uses the outputs from the shadow models as the features to train an audit model. Subsequent work develop solutions under various settings, including speech recognition models [11, 48], metric learning models [12, 38] and large language models [32, 47].

A common thread of existing user-level MI methods is to aggregate the information across multiple samples (e.g., different voices of the same speaker) to perform MI [11, 12, 32, 58]. The MI process in *MembershipTracker* follows a similar philosophy. However, prior work often requires access to additional reference models [32] or shadow models [11, 12, 58], and they still have limited MI effectiveness (e.g., limited TPR under the low FPR regime [32, 47]). *MembershipTracker* overcomes these limitations by means of a novel data-marking and specialized MI auditing process, which can simultaneously achieve high TPR and low FPR for data provenance.

Concurrent work by Huang et al. [31] proposes a general framework for auditing unauthorized data use in training DL models. Their idea is to generate two perturbed copies of the target data, randomly publish one of them, and then compare the model's membership score on the published vs. the unpublished one. While their framework can work for different domains such as foundation models, they still require the data holders to mark a substantial portion (1%∼10%) of dataset. Instead, *MembershipTracker* considers the more challenging (and realistic) setting where the marked data amounts to $\leq 0.1\%$ of the dataset. We also "stress test" their technique under a similar setting as ours and find that their performance degrades considerably then (see Appendix F).

3 Problem Formulation

Membership Inference (MI) Game. We denote $F_\theta : \mathcal{X} \to [0,1]^n$ as a model that maps an input sample $x \in \mathcal{X}$ to a probability vector over n classes. $D = \{(x_i, y_i)\}_{i=1}^n$ is a training set sampled from some distribution $\mathcal{D}$. $F_\theta \leftarrow \mathcal{T}(D)$ denotes a model F_θ produced from running the training algorithm $\mathcal{T}$ on D. Next, we define a MI game, which proceeds between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger samples a dataset $D \leftarrow \mathcal{D}$, and a target point $z \leftarrow \pi$ (such that $D \cap \pi = \emptyset$).
2. The challenger trains a model $F_\theta \leftarrow \mathcal{T}(D \cup \{z\})$ on the dataset D and target point z.
3. The challenger gives the adversary query access to F_θ.
4. The adversary emits a guess $\hat{z} \in \pi$.
5. The adversary wins the game if $\hat{z} = z$.

For a specific target point x, $\pi = \{x, \perp\}$, where $\perp$ indicates the absence of an example. The adversary's goal is to guess whether the model F is trained on D or $D \cup \{x\}$.

3.1 MI for Data Provenance

We now describe the MI game for data provenance, which proceeds between a challenger (model creator) $\mathcal{C}$ and a target user U. We consider image classification in this work, and

U seeks to audit whether his/her images (denoted as X) are used by $\mathcal{C}$ to train a model F without permission. We add the ability for U to modify X for provenance purpose.

1. The user is allowed to modify $X \in \pi$ ($\pi = \{X, \perp\}$).
2. The challenger samples a dataset $D \leftarrow \mathcal{D}$, and a target set $Z \leftarrow \pi$ (such that $D \cap \pi = \emptyset$).
3. The challenger trains a model $F_\theta \leftarrow \mathcal{T}(D \cup \{Z\})$ on the dataset D and target set Z.
4. The challenger gives the user query access to F_θ.
5. The user emits a guess $\hat{Z} \in \pi$.
6. The user wins the game if $\hat{Z} = Z$.

For a given user U, $\pi = \{X, \perp\}$, where $\perp$ indicates the absence of the target samples by U, and the goal is to determine whether F is trained on D or $D \cup \{X\}$.

3.1.1 User capability

We consider ordinary personal users U with limited ML expertise and resources, and they are willing to add visual distortion to their data for provenance purposes. We next specify their capabilities and constraints.

First, we assume U can only mark a small number of samples, and these are to be included as part of a larger dataset. This simulates the realistic scenario where the dataset is curated from multiple data sources (e.g., [19]), and thus data contributed by each user constitutes only a small portion of the dataset (e.g., 0.1%).

Next, in the provenance verification process, we assume U has access to an auxiliary set of non-member samples, which is a common assumption to control the MI process at a low FPR regime [6, 8, 70]. U also has black-box access to query the target model. In a real world setting, users often have limited control and knowledge beyond their own data, and thus we specify the constraints by U as follows.

- U cannot modify their data's labels, which are assigned by the model owner or an external service [19, 21, 68].
- U has no access to D (i.e., the training data by other users).
- U has no knowledge of the black-box model F.
- U has no expertise/resources to train any shadow/proxy models, as in many settings, training even a single shadow model can impose prohibitively high data and compute requirements (especially for the large models) [65].

3.1.2 Evaluation metric

We follow the best practice to measure the MI success by its true positive rate (TPR) and false positive rate (FPR) [8]. There are two ways to define the success metric: ① TPR under a fixed FPR; or ② FPR under a fixed TPR. We use metric

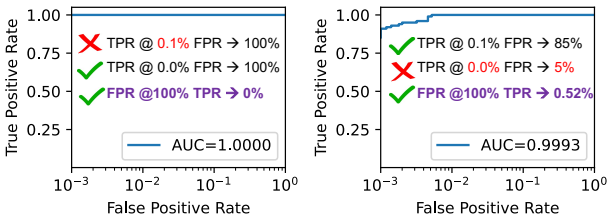


Figure 2: Comparing two metrics for evaluating MI success in our work: ① TPR@fixed FPR; ② FPR@fixed TPR. In ①, the pre-defined FPR threshold can be too conservative (✗ on the left) or too aggressive (✗ on the right). Instead, MI for *data provenance* requires both low FPR and high TPR, for which ② is a more suitable evaluation metric.

②, and we first explain why ① is unsuitable for evaluating the data-provenance based MI method in our work.

Motivation. Metric ① is commonly adopted by existing work where MI is posed as an attack to expose privacy leakage in DL models [6, 8, 70]. Their goal is to evaluate whether the adversary can reliably de-identify (even just a few of) the training samples. This places emphasis on the requirement of low FPR, for which ① represents a suitable metric [6, 8, 70].

In comparison, in our work of tracing data provenance, *both* low FPR (to avoid false accusation) *and* high TPR (for correct data provenance) are crucial. However, manually choosing a low FPR threshold and evaluating the corresponding TPR (as in ①) can be undesirable, as the pre-defined FPR threshold may be too conservative or aggressive. We use Fig. 2 (from our experiment) to explain next.

For the left method in Fig. 2, using a 0.1% FPR threshold is too *conservative*: this method can yield the same 100% TPR under a lower FPR of 0%. Setting a 0% FPR threshold is appropriate for the left method, but this becomes too *aggressive* for the right method in Fig. 2: this other high-power method (with 0.9993 AUC) yields a poor outcome of 5% TPR.

Proposed metric. To overcome the above issue, we advocate the use of metric ② (FPR under a fixed TPR).

Under a 100% TPR, the left method in Fig. 2 has a 0% FPR, and the right method has a slightly higher 0.52% FPR. Both results are able to appropriately reflect the two MI methods' capability in fulfilling the requirement of low FPR (0% and 0.52%)[1] and high TPR (100%) for tracing data provenance.

Based on the above, we adopt FPR@fixed TPR as the evaluation metric in this work, and we set a 100% TPR threshold to simulate the case where all the users whose data are misused without their consent can trace the provenance of their data (see our design goal in Section 4 next). An ideal technique should incur as low FPR as possible.

[1]Note that this metric serves only to evaluate the MI success, and the users can still choose an arbitrary FPR threshold (e.g., 0% or 1% FPR) to control the MI process when auditing their actual data (Section 4.4.3).

4 Methodology

We first describe our design goals, and then conduct an empirical study to understand why directly leveraging existing MI methods cannot meet our goals. We then present *MembershipTracker*, a two-step technique with a data marking and provenance verification process: first by marking the users' protected data with targeted changes (Section 4.3); then performing a specialized MI process to audit whether the marked data are used to train the model (Section 4.4).

Design Goals. There are three design goals in our work.

Goal (1): *High provenance effectiveness.* Our main goal is to enable those users whose data are misused for model training to perform successful data provenance (100% TPR), and we also want to ensure low FPR to avoid false accusation.

Goal (2): *Preserving high visual quality.* The modification created to mark the users' data should not severely distort their visual information, e.g., the identity in a facial image should still be easily recognizable.

Goal (3): *Minimal technical requirements.* The provenance tool should be usable by ordinary users even with limited expertise and resources (specified in Section 3.1.1).

4.1 Motivation

Our work addresses data provenance through MI, and there are today a number of available MI methods [6, 8, 65, 70]. However, these methods still suffer from limited MI effectiveness and/or impose strong assumptions on the users - we elaborate both limitations next.

① Existing MI methods have limited provenance effectiveness, and incur very high FPR under 100% TPR.

To validate this, we use two methods in the state-of-the-art Likelihood-ratio attack, one is with and the other without shadow model calibration (128 shadow models) [8]. We evaluate both methods on a WideResNet-28-4 model trained on half of the training set in CIFAR100 dataset, and they both incur a high FPR of 75.32% and 64.62%, under 100% TPR.

With that said, the above results should *not* be interpreted as the failure of these approaches, because they were originally built from an adversarial perspective to expose the worst-case privacy leakage in DL models, where even just reliably de-identifying a few member samples is considered meaningful [8], e.g., the shadow-model based method in Fig. 3 can achieve 27.21% TPR@0.1% FPR.

On the other hand, our work considers MI for *data provenance* purposes, which represents a more challenging task, and requires both low FPR and high TPR. The high TPR is crucial for the users to be able to detect the unauthorized use of their data, while low FPR is important to minimize the chance of false accusation.

As in Fig. 3, existing MI methods still struggle with this task. The reason is that the training samples are not always strongly memorized by the model (except some outliers) [8, 65, 71], which is directly related to the ability of performing accurate MI. This problem motivates related work [10, 13, 65] to amplify the model's memorization on the training samples, and we discuss their limitation next.

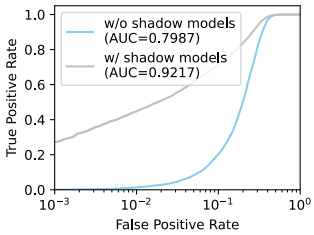


Figure 3: MI for tracing data provenance requires *both* high TPR (for correct provenance detection), *and* low FPR (to avoid false accusation). However, even state-of-the-art MI methods [8] cannot meet this challenging goal, and are unsuitable for data provenance purposes.

② State-of-the-art MI methods either require training shadow models or assume control to manipulate the data labels, both are beyond ordinary users' capabilities.

For example, leading MI methods [6, 8, 70] commonly require training shadow models to calibrate the inference process, which assumes additional data and compute access, and is challenging for the non-expert users.

There are other methods that do not require shadow-model calibration, but they either suffer from poor effectiveness [8, 71] or impose unrealistic assumption on the users [10, 65]. For instance, the approach by Tramer et al. [65] can improve the MI success without shadow-model calibration, but it instead assumes the users have the ability to mislabel some training data, which can be challenging in the cases where the users have no control over the data labels [19, 21, 68]. Even if this is feasible, we find that the increase of MI success is still largely limited (see Appendix E).

4.2 Overview

We next present *MembershipTracker*, a technique that requires neither shadow-model training nor manipulating data labels, and can still achieve high-power MI (with low FPR and high TPR) for tracing data provenance. It consists of a data marking and a MI-based verification process - we explain them next.

4.3 Data Marking

The goal of data marking is to ensure that the membership of the marked samples can be reliably de-identified. To this end, we first observe that the ability to perform accurate MI is directly related to the model's propensity in memorizing individual data points, and data that are strongly memorized by the model (e.g., outlier samples, mislabeled samples) are known to be easier to be de-identified [8, 14, 65, 71]. Based on this, our goal can be formulated as: how can the users modify

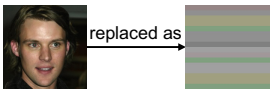
their data to amplify the model's memorization on the target data while preserving high visual quality?

In Section 4.3.1, we first present an initial approach that is highly effective in inducing the target samples to be memorized by the model, but it also completely destroys the visual information in the data. We use this effective yet unrealistic method as a strawman approach, and then introduce our proposal of a two-step solution, which can greatly reduce the visual distortion to the data while still fulfilling our goal.

4.3.1 An initial strawman approach

Inspired by the observation that atypical samples presented in the dataset are easier to be de-identified (e.g., mislabeled samples [65], or samples that are out-of-distribution - OOD [8]), a straightforward approach is to directly *replace* the original target samples as OOD samples.

There are different ways to generate OOD samples, and we develop a simple method that creates samples consisting of random color stripes, such as the one shown below (the method's details are deferred to Section 5). Meanwhile, other alternative methods such as using samples from an OOD dataset may also be considered (Appendix C.1.3).



We first conduct an experiment to validate the effectiveness of the above approach. We use the CIFAR100 dataset with half of the training set for model training. We consider 100 target users from different classes replacing their data as some random OOD samples, and each contributes a small number of samples to the training set (0.1% in proportion). In terms of the signal function for MI, we follow prior work [54, 70] to use the prediction loss values.

To perform MI, we compare the prediction loss on the target OOD samples (from the member users) and some non-target OOD samples (from the non-member users). We find that the model indeed strongly memorizes those OOD samples present in the training set, for which we observe a 0% FPR@100% TPR. In comparison, if the target data are unmodified, the model only exhibits weak memorization on them, which yields 56.32% FPR@100% TPR.

Limitation. Despite its effectiveness, this approach completely removes the visual information in the data (e.g., the facial identity is no longer recognizable), and they are also easy to notice. Next, we present two methods that can greatly reduce the visual distortion to the target data, while still enabling them to be strongly memorized by the model (Fig. 4).

4.3.2 Step 1: Image blending

As discussed earlier, the marked samples should contain the original feature (to preserve visual information), as well as the OOD feature (for provenance purposes). Inspired by the common image blending technique [68, 74], we propose to blend the target sample with the OOD feature:

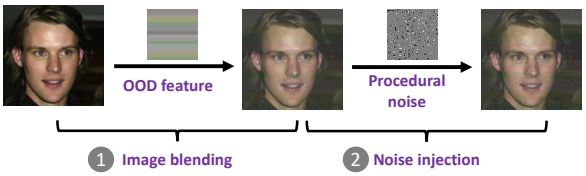


Figure 4: The two-step data marking process: (1) blend the original samples with OOD feature; (2) inject procedural noise. These subtle changes can induce the target samples to be strongly memorized by the model trained on them.

$$x \oplus (x_{ood}, m) = m \cdot x + (1 - m) \cdot x_{ood}, \quad (1)$$

where m moderates the contribution of different features.

By using a large m, we can largely preserve the high image quality (e.g., the center image in Fig. 4 is marked with $m = 0.7$) while keeping the OOD feature to amplify the model's memorization on the resulting samples. We find that merely blending the OOD feature into the target samples ($m = 0.7$) can reduce the FPR@100% TPR from 56.32% to 23.58%.

Limitation. Although the FPR is largely reduced, it is still too high. We find that this is due to the existence of the original feature, which hinders the model's memorization on the OOD feature. We next explain how to mitigate this.

4.3.3 Step 2: Noise injection

Our idea is to inject a small amount of noise to suppress the influence of the original feature, and have the model memorize the OOD feature.

For this, we draw inspiration from the concept of *adversarial samples*, which works by adding imperceptible noise to the inputs and cause them to be misclassified by the model [17, 40, 43]. The perturbation can thus be viewed as the noise that can suppress the influence of the original feature and cause the model to predict the wrong label.

In our context, we can inject adversarial perturbation into the users' target samples and prompt the model to memorize the OOD feature. There are different ways to generate the perturbation. Optimization-based approach requires access to some proxy model to generate the perturbation [40, 43], which can still be challenging for the ordinary users we consider.

We therefore resort to another optimization-free strategy that does not assume any additional access. In particular, Co et al. [17] propose that *procedural noise functions* can be used to generate input-agnostic adversarial perturbation. These functions are commonly used in computer graphics to generate different textures and patterns and enrich the image details [17]. Co et al. find that the procedural noise patterns share visual resemblance with existing adversarial perturbation patterns, and demonstrate that they can be similarly used to construct adversarial samples.

This inspires our solution in injecting *perlin noise* [17] to the target samples, and its generation is as follows.

$$S_{perlin}(x,y) = \sum_{n=1}^{\Omega} p(x \cdot \frac{2^{n-1}}{\lambda_x}, y \cdot \frac{2^{n-1}}{\lambda_y}), \quad (2)$$

$$G_{perlin}(x,y) = sin((S_{perlin}(x,y)) \cdot 2\pi\phi_{sine}), \quad (3)$$

where $\lambda_x, \lambda_y, \Omega, \phi_{sine}$ are for controlling the noise value at point (x,y). λ_x, λ_y are the wavelengths, Ω is the number of octaves that contribute to the most visual change, ϕ_{sine} is the periodicity of the sine function that creates distinct bands in the image to achieve a high frequency of edges, which can achieve more distinct visual patterns [17]. We follow Co et al. to use the L_∞ norm as the perturbation budget δ and create random perlin noise [17] for data marking.

Generating the perlin noise is a lightweight process that does not assume access to a proxy model. Despite its simplicity, we find that injecting a small amount of perlin noise (with $\delta = 8/255$) is very effective and it can further decrease the FPR from 23.58% to 7.97% (a 66% reduction).

Overall, the two-step marking process is able to reduce the FPR from 56.32% to 7.97%. While the FPR can be further reduced by using a smaller m for image blending or larger δ for noise injection, it would undesirably increase the visual distortion (see Appendix C.1.2 for an evaluation). We thus propose another innovation in the MI process to mitigate the FPR, and without degrading the image quality.

4.4 Data Provenance via Membership Inference

To start with, we note that so far we have been considering MI on a common *instance* basis [6, 8, 56, 70], which uses the per-instance loss as the signal function value for MI. We first explain why this approach would incur high FPR, and then present a specialized MI process to overcome it.

4.4.1 Why instance-based MI suffers from high FPR

It uses the per-sample loss for MI, which can incur high FPR even if the prediction loss on the target marked samples are very low. This is because there may exist many non-member samples with low loss values as well, which would be misidentified as member samples, and result in high FPR.

For instance, the left figure in Fig. 5 shows the individual loss values on different samples, where many non-member samples also yield low prediction loss (the low loss values are shown towards the right side because they are plotted in the logit-scaled form [8] for visualization purposes). This consequently leads to a 7.97% FPR @100% TPR.

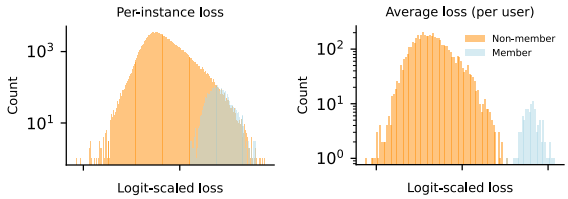


Figure 5: Comparison of using the per-instance loss and the average loss across the samples by each user (each contributes 0.1% of the training data) as the signal function for MI verification. The former yields 7.97% FPR@100% TPR; while the latter has 0% FPR@100% TPR (our proposal).

4.4.2 Set-based membership inference for reliable data provenance

To overcome the above, our key insight is that, even though each target user only contributes a small number of samples to the model's training set (e.g., 25 out of 25,000 instances), they can still leverage the *collective* information across their own samples to construct a more reliable signal function for MI, thereby reducing the FPR.

To realize this, we start by identifying a pattern that can distinguish the model's behavior on the member and non-member samples, and it is as follows. For the *non-member samples*, although a few of them may have low loss, the majority of them would incur much higher prediction loss, because these samples are not used for training the model (e.g., see the left of Fig. 5). In contrast, the *member samples* predominantly have very low prediction loss.

Therefore, we make the observation that *the average loss for the samples by the non-member users are higher than that by the member users*. A visualization for this is on the right of Fig. 5. This leads to our proposal of the *set-based* MI process, which leverages the *average loss* of the small set of target samples by each user as the signal function for MI. We explain the verification process next.

4.4.3 Provenance verification process

Algorithm 1 outlines the process. Let each user possess k samples, where k is a small number compared with the dataset size (e.g., 25 out of 25,000 samples). $\mathcal{P}_{\text{out}}$ denotes a set of non-member data to control the MI process at the low FPR regime (Section 3.1.1), which are also marked with some random outlier features and perlin noise (data-marking details in Section 5). By comparing the model's behavior on the target data and the data in $\mathcal{P}_{\text{out}}$, the auditing user U is to detect whether his/her target data are used to train F.

The first step is to derive an inference threshold c based on a pre-defined FPR (α). α is usually a small number such as 0.1% or 0% to ensure low false positive (e.g., Section 5 shows that *MembershipTracker* can support the users to achieve 100% TPR with 0% FPR in many cases). To derive the threshold c, U first computes the empirical loss histogram for data in

Algorithm 1 Set-based MI verification process

Input: F: The target DL model;
U: The auditing user (each user with k target samples);
$\mathcal{P}_{\text{out}}$: Users from the out world (i.e., non-member data);
α: FPR to control the verification process (e.g., 0.1%);
$\mathcal{L}(F,(x,y))$: Loss function (e.g., cross-entropy loss);

1: **function** DATA PROVENANCE($F, U, \mathcal{P}_{\text{out}}, \alpha, \mathcal{L}$)
2: $\forall U_i \in \mathcal{P}_{\text{out}}$, compute $\mathcal{L}(F,U_i)$, where $U_i = \{(x_j^i, y_j^i)\}_{j=1}^k$
3: $\mathcal{L}_{\text{avg}} \leftarrow \{\overline{\mathcal{L}(F,U_1)}, \cdots, \overline{\mathcal{L}(F,U_n)}\}, n = |\mathcal{P}_{\text{out}}|$
4: $c \leftarrow \alpha$-percentile of the CDF for the loss histogram of $\mathcal{L}_{\text{avg}}$
5: /* Data provenance based on the inference threshold c */
6: Compute $\mathcal{L}(F,U)$, where $U = \{(x_j, y_j)\}_{j=1}^k$
7: $\mathcal{L}_{\text{avg}}^U \leftarrow \overline{\mathcal{L}(F,U)}$
8: **if** $\mathcal{L}_{\text{avg}}^U < c$ **then**
9: U's data are used to train F (i.e., member)
10: **else**
11: U's data are not used to train F (i.e., non-member)
12: **end if**
13: **end function**

the out world $\mathcal{P}_{\text{out}}$. For each non-member user in $\mathcal{P}_{\text{out}}$, the auditing user U first computes the per-instance loss on their data (line 2), and then derives the average loss for each (line 3). Line 4 estimates the CDF for the loss distribution, and computes its α-percentile to derive a threshold c [70].

Next, the auditing user U similarly computes the average loss on his/her samples, and compares it against the MI threshold c. If the loss is lower than c, the user's data are deemed to be used for training the model (vice versa).

5 Evaluation

Datasets and model training. We consider six common benchmark datasets, including CIFAR100 [35], CIFAR10 [35], ArtBench [41], CelebA [45], TinyImageNet [36] and ImageNet-1k [19]. This section considers the first five datasets and the evaluation on the ImageNet-1k dataset is in Section 6. We follow standard model training practice such as using data augmentations and validation set - the model training details are in Appendix A.

***MembershipTracker* configuration**. We explain the default configurations for the main experiments, and we study different configurations in the ablation study (Appendix C).

As explained earlier, we create outlier features as random color stripes, each with 16 stripes from 11 different common colors. Such a large feature set (11^{16} choices) can support multiple users to generate their own features for data marking, and we use a image blending ratio $m = 0.7$. Next, we follow Co et al. [17] to generate random perlin noise for each sample with $\delta = 8/255$. We provide additional visualizations of the original and marked samples in Appendix A.

We assume each user possesses a small number of samples from a given class, which constitute 0.1% of the training set (the detailed dataset sizes are in Appendix A).

As explained in Section 3.1.2, we adopt FPR@100% TPR as the evaluation metric, and we use 5,000 non-member users for computing FPR. We use all the samples that are not used for training as the non-member set. Due to the limited size of the non-member set, the non-member users' samples are drawn randomly from the non-member set (with replacement), and these are similarly marked with random outlier features and perlin noise.

Table 1: Evaluation on different datasets (parenthesized numbers show accuracy diff. compared with the original models).

Dataset	FPR@100% TPR	Model accuracy
CIFAR100	0.00	66.34 (-0.41)
CIFAR10	0.00	90.64 (-0.50)
CelebA	0.00	69.41 (-0.63)
ArtBench	0.00	59.46 (-0.42)
TinyImageNet	0.00	67.22 (-0.40)

To perform MI, we directly use the cross-entropy loss as the signal function value, which is easy to compute by the users. We consider the global-threshold method by Carlini et al. [8] as the baseline method; and we do not choose the shadow-model-based approaches [6, 8, 70]. The reason is that, even though these methods can achieve better results, they still yield limited increase of MI success (Section 4.1); and more importantly, training shadow models is prohibitively challenging for ordinary personal users to carry out (Section 3.1.1).

The closest work to ours in improving the MI success without shadow-model calibration is Tramer et al. [65]. However, their technique assumes the users can manipulate the data labels, which excludes the real-world scenarios where the data labels are assigned by the model creator or an external service [19,21,68]. Though *MembershipTracker* does not impose such an assumption, for completeness, we still quantitatively compare it with the method by Tramer et al. (Appendix E).

5.1 Results

We first evaluate the single-target-user setting, and then the multi-target setting.

Single-target evaluation. For each experiment, we randomly choose 5 target users from different classes and report the average results. Table 1 presents the results.

As shown, with *MembershipTracker*, the users can reliably trace the provenance of their marked data with 0% FPR@100% TPR. In comparison, without *MembershipTracker*, performing standard instance-based MI on the unmodified target samples incurs much higher FPR, with an average of 47.52% FPR@100% TPR. This is because: 1) the unmarked samples cannot be strongly memorized by the model; and 2) the instance-based MI process fails to leverage the collective information across the target samples by each user. These two limitations are overcome by *MembershipTracker*'s data marking and set-based MI process; and the detailed ablation study on these two components is presented in Appendix C.

Table 2: Evaluation on multiple target users. Supporting more target users indicates more specially-marked samples in the training set, which can lead to higher accuracy drop; but *MembershipTracker* still maintains high auditing performance.

Num of target users	20	50	100	200	300	500
FPR@100% TPR	0.00	0.00	0.00	0.00	0.00	0.00
Accuracy drop	-0.44	-0.56	-1.39	-2.32	-4.47	-8.93

Meanwhile, since the target samples only occupy a small fraction of the training set (0.1%), the models still maintain high performance, and have $<$1% accuracy difference compared with the ones trained without *MembershipTracker*.

Evaluation on different architectures and training-set sizes. We next consider additional evaluation (using CIFAR100 dataset) on different model architectures (including WideResNet [72], ResNet [23], ResNext [69], SeNet [28], GoogleNet [63] and DenseNet [29]), as well as different training-set sizes (from 5,000 to 25,000). We observe a similar trend as before (thus we omit the discussion), and the detailed results are in Appendix B.

Multi-target evaluation We now evaluate how *MembershipTracker* can support multiple users simultaneously (using CIFAR100 dataset), and the results are in Table 2. We study a large number of target users (more than the number of classes in the dataset), which encompasses the settings where the users are from different classes and from the same class.

Supporting multiple users requires the model to memorize more target samples (from different users). However, DL models are known to be capable of memorizing a large corpus of data (e.g., to encode sensitive information [14, 57]). Likewise, in our case, the model still strongly memorizes the marked samples and enables *MembershipTracker* to achieve 0% FPR@100% TPR even under multiple target users.

Meanwhile, as the number of target users increases, the model also has higher accuracy drop. For instance, supporting 100 target users (i.e., 10% of training data are marked) incurs a 1.39% accuracy drop, while supporting 500 users has a higher 8.93% drop. This is understandable as supporting more users implies more training samples are marked.

5.2 Robustness to Countermeasures

While Section 5.1 demonstrates *MembershipTracker*'s capability in enabling the users to perform reliable data provenance, a deliberate model creator can also employ countermeasures to sabotage *MembershipTracker*.

To understand this, we evaluate a total of 6 types of countermeasures in the model, input and output level (Table 3), and we use the CIFAR100 dataset. For the model-level defenses that require model training, we first consider the single-target setting (similar to Section 5.1), and then the multi-target setting. For the input- and output-level defenses, we consider the more challenging multi-target setting (100 targets in total).

Table 3: Overview of the evaluated countermeasures.

Defense level	Defense type
Model level	① MI defense; ② Adversarial augmentation; ③ Model fine-tuning; ④ Model pruning
Input level	⑤ Out-of-distribution and spurious features detection
Output level	⑥ Add noise to the output, or return only the label

Table 4: Evaluating *MembershipTracker* against DP-SGD with different amounts of noise injected.

Noise multiplier σ	FPR@100% TPR	Model accuracy
w/o defense	0.00	66.34
0.01	0.01	60.10
0.03	0.02	57.26
0.05	0.38	54.70
0.2	1.07	51.56
0.3	3.10	50.03

5.2.1 MI defense

We first evaluate several representative privacy defenses for mitigating MI (five in total).

DP-based defense. Differentially private (DP) training [3] is a principled defense that can bound the influence of any training samples to the model, by means of gradient clipping and noise injection to the clipped gradients. We adopt the implementation from Aerni et al. to consider the strong DP-SGD baselines that comprise various DP-training techniques such as custom initialization scheme, augmentation multiplicity [4]. We also follow their work to tune hyperparameters for high model utility. We use a tight clipping norm of 1, and inject different amounts of noise (σ) in the evaluation.

The results are in Table 4. DPSGD with a loose DP bound is known to be able to reduce the exposure from the *instance-based* MI methods that aim to de-identify some individual training instances [4, 8], and our results also validate this (e.g., in our context, the FPR@100% TPR under instance-based MI is increased to 47.22%, with $\sigma = 0.01$). With that said, *MembershipTracker* can still overcome the defense via its *set-based* MI process, because the *average loss* of the samples by the member users, though increased due to defense, are still lower than that by the non-member users (average 0.1642 vs. 4.130). This avails *MembershipTracker* to maintain a low 0.01% FPR@100% TPR (and the defense already incurs 6.24% accuracy drop). Increasing the amount of noise can further increase the FPR by *MembershipTracker*, but it also inflicts a larger accuracy drop (undesirable).

Empirical defense. Next, we also evaluate two representative defenses that aim to provide strong empirical privacy while preserving model utility: SELENA (USENIX'22) [64] and HAMP (NDSS'24) [15]. We follow the original work to set up both techniques. They both can mitigate the model's memorization (increase the prediction loss) on the marked samples, and with small accuracy drop (2.17% and 1.42%). However, we find that the average loss of the samples by

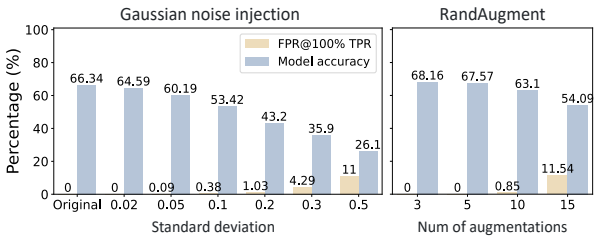


Figure 6: Adv. augmentation by Gaussian noise injection and RandAugment [18]. *MembershipTracker* is resilient to moderate input augmentation, and aggressive augmentation can increase the TPR, but also jeopardize model accuracy.

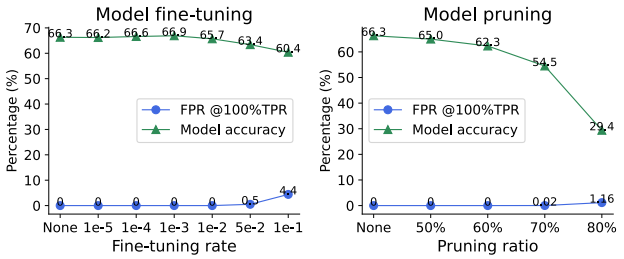


Figure 7: Defense evaluation on model fine-tuning and model pruning. *MembershipTracker* is resilient to moderate model fine-tuning and pruning. Aggressive countermeasures can slightly increase the FPR incurred by *MembershipTracker*, but at the cost of major accuracy degradation.

the member users are still much lower than that by the non-member users, and thus *MembershipTracker* still achieves a low FPR of 0.97% (on SELENA) and 0.83% (on HAMP).

Finally, we evaluate two common regularization techniques: *early stopping* and *dropout*. We find that *MembershipTracker* can still achieve $<0.1\%$FPR@100% TPR, *unless* under excessive regularization (e.g., aggressively early-stopping the training or using a large dropout rate). However, this would also cause a large ($>10\%$) accuracy drop to the model.

5.2.2 Adversarial augmentation

We consider employing additional augmentation techniques during training to mitigate the model's memorization on the marked samples. We study two strategies, one by injecting Gaussian noise and the other by performing multiple random data augmentations (RandAugment [18]). For the former, we inject zero-mean noise with various standard deviation values ($0.02\sim0.5$); For RandAugment [18], we consider different number of augmentations ($3\sim15$).

When the samples are moderately augmented, in the form of Gaussian noise or multiple random augmentations, *MembershipTracker* still enables the target users to reliably trace the provenance of their data with low FPR.

Aggressive augmentation such as applying 15 random transformations to each sample can further increase the FPR to 11.54%, but it also reduces the accuracy from 66.34% to 54.09%, as such aggressive augmentation severely degrades the image quality and leads to inferior model performance.

5.2.3 Model fine-tuning

We now evaluate how model fine-tuning can mitigate the model's memorization on the target samples. We assume an ideal defender with access to a small set (equivalent to 10% of the training set) of clean and unmodified samples. We fine-tune the model with various fine-tuning rates from $0.1\sim0.00001$, with 20 epochs for each. The results are shown on the left of Fig. 7. When the fine-tuning rates are small (<0.05), the models can preserve high model accuracy, yet they still exhibit strong memorization on the target samples, and *MembershipTracker* still achieves 0% FPR@100% TPR.

Using a larger fine-tuning rate (e.g., 0.05, 0.1) can update the model in a more aggressive manner and hence alleviate the model's memorization on the target samples. This, however, also hurts the model's accuracy. For a fine-tuning rate of 0.05, the model accuracy is reduced by 2.9%, and the FPR by *MembershipTracker* increases from 0% to 0.5%. A larger fine-tuning rate of 0.1 can further increase the FPR to 4.4%, but it also causes a 5.97% accuracy drop.

5.2.4 Model pruning

Another related countermeasure we consider is to prune the model's parameters, with the goal of removing the parameters that are associated with the target samples. We use the popular pruning strategy by Han et al. [22], which sets the parameters with small absolute values to zero and has minimal impact to the model's performance. We evaluate different pruning ratios and the results are on the right of Fig. 7.

We find that *MembershipTracker* is highly resilient even when 70% of the parameters are pruned, where *MembershipTracker* has only 0.02% FPR@100% TPR, but the defense already incurs a 11.8% accuracy drop. More aggressive pruning with a 80% pruning ratio can slightly increase the FPR to 1.16%, but with a much higher 36.9% accuracy drop.

5.2.5 Model-level defenses under multi-target setting

The previous model-level defenses are evaluated under the single-target setting. We now evaluate a more challenging setting where *MembershipTracker* needs to support multiple users at the same time. We consider 25 target users, and summarize the key findings below.

Among different defenses, we find that *model fine-tuning* stands out as the most potent solution that outperforms other countermeasures in terms of high mitigation performance with low accuracy drop.

Compared with other methods (such as specialized MI defenses [3, 15, 64]), fine-tuning only needs to perform standard

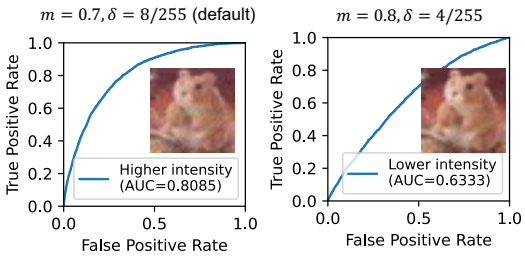


Figure 8: Unsupervised OOD detection. How different data-marking intensities affect the detection performance. *Left*: Samples marked with higher marking intensity are prone to be detected (0.8085 AUC). *Right*: Reducing the marking intensity can effectively reduce the AUC to only 0.6333.

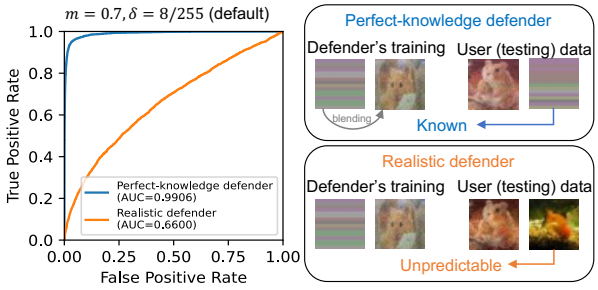


Figure 9: Supervised OOD detection. *Blue* line assumes a defender with *perfect* knowledge of the type of outlier features adopted by the users for data marking. *Orange* line considers a *realistic* defender with knowledge of the possible types of outlier features, but the exact one followed by the users to mark their data is unpredictable.

fine-tuning on a small set of samples, and it can increase the FPR@100% TPR to 21.4% (with a fine-tuning rate of 0.05) with a mere 2.09% accuracy drop.

Hence, unlike the single-target setting where *MembershipTracker* can maintain high performance, supporting multiple target users under model-level defenses remains a major challenge by *MembershipTracker*. We leave the potential improvement of this aspect to future investigation.

5.2.6 Input detection

Given that *MembershipTracker* leverages outlier features and perlin noise to the mark the target data, the defender may also perform *out-of-distribution (OOD) detection* or *spurious features detection* on the specially-marked samples. We consider both types of countermeasures and we discuss them next.

Unsupervised OOD detection. We consider eight common detection techniques: Mahalanobis [37], MSP [25], Energy-based OOD [42], RMD [51], kNN [62], KLMatching [24], VIM [67], DICE [61]. We follow the implementation from [34], and use a pre-trained ImageNet model to produce the feature representation. As before, we assume an ideal defender with access to a small set of clean and unmodified samples (equivalent to 10% of the training set) for fitting the unsupervised detector, which does not assume any prior knowledge of the potential outlier samples.

We report the results on the left of Fig. 8, where, for brevity, we report only the highest AUC among all detectors. We find that the Mahalanobis-based approach [37] achieves the highest performance with 0.8085 AUC. In light of this, we next explain how to (partially) mitigate it.

Our idea is to reduce the amount of distortion applied to the data. For this, we conduct an experiment to reduce the intensity of the outlier features (by increasing the image blending ratio m from 0.7 to 0.8) and perlin noise (by reducing the perturbation budget δ from 8/255 to 4/255). The results are shown on the right of Fig. 8, where the highest AUC is reduced from 0.8085 to 0.6333, which translates to a 57% reduction in the detection performance (over random guessing).

Naturally, reducing the marking intensity would attenuate the model's memorization on the marked sample, and increase their prediction loss. But we find that the prediction loss on the member samples are still much lower than that on the non-members, and thus *MembershipTracker* can overcome it via the set-based MI process (similar to Section 5.2.1), and maintain a 0% FPR@100% TPR[2].

Supervised OOD detection. Next, we consider a more knowledgeable defender who is aware of the potential data marking strategy by *MembershipTracker*. We use the MCHAD method to train a supervised OOD detector [33], and we consider two separate scenarios below.

(1) We first assume a hypothetical defender with *perfect* knowledge of the data marking strategy adopted by the users, i.e., each sample is marked with features consisting of random color stripes and perlin noise. The result is in the blue line in Fig. 9. As expected, the detector is very capable of detecting the samples that are marked with the same strategy as the one used for training the detector itself (e.g., see the top right illustration in Fig. 9).

Though this represents the worst-case scenario against *MembershipTracker*, it may be difficult to realize in practice, as the outlier features created by the users for data marking can take *diverse* forms (e.g., as features consisting of random color stripes, or features from an OOD dataset - details next), and the exact one used by the users can be hard to predict.

(2) Hence, a more realistic setting is to consider a defender with knowledge of the possible types of outlier features, but is not aware of the exact one adopted by the users to mark their data. In this case, the outlier features in the samples used for training the detector are not in the same type as those in users' actual data (e.g., see the bottom right illustration in Fig. 9).

Specifically, we train the detector on samples marked with random color stripes, and evaluate it on samples marked

[2]However, there are certain scenarios where the reduced data-marking intensity would lead to lower auditing performance, and an evaluation for this is in Appendix C.1.2.

with the features from an OOD (TinyImageNet) dataset (this is a similarly effective alternative strategy we study in Appendix C.1.3). The result is in the orange line in Fig. 9, and the detection performance degrades considerably under such a realistic setting (with only 0.66 AUC).

Although the defender can attempt to train the detector on samples marked with different types of outlier patterns, with the hope that some of them may overlap with those in users' data, this essentially turns the detection into a cat-and-mouse game, and may not be a desirable practice.

Spurious features detection is another related method. We evaluate a leading technique by Neuhaus et al. [49] based on neural PCA components. But we find that it also has limited success in detecting the marked samples (0.6181 AUC).

5.2.7 Output perturbation

MembershipTracker leverages the model's outputs to perform MI auditing, and we consider two strategies to perturb the model's outputs: (1) injecting Gaussian noise to the outputs; and (2) returning only the top-1 prediction label [16]. However, we find that *MembershipTracker* still maintains high auditing success even with the perturbed outputs.

For instance, aggressively injecting Gaussian noise (with zero mean and standard deviation $\sigma = 3$) degrades the accuracy from 65.32% to 52.05%, under which *MembershipTracker* still maintains 0% FPR@100% TPR. Completely shielding the output probabilities is ineffective either, and *MembershipTracker* can still achieve 0.04% FPR@100% TPR in the label-only setting. The detailed evaluation can be referred to Appendix G.

6 Evaluation on ImageNet Training

This section evaluates *MembershipTracker* under the full-sized ImageNet-1k training (over 1.28 million training samples). We train a ResNet50 [23] and Swin-Transformer [44], and they have 75.09% and 80.29% accuracy respectively (both with <1% accuracy drop). We consider the challenging setting of supporting multiple users in different classes (1,000 target users in total). Each user contributes 125 samples to the training set, which is <0.01% in proportion.

We find that even under the large-scale training setting, the marked samples can still be strongly memorized by the models, which enables *MembershipTracker* to achieve 0% FPR@100% TPR for all the users.

Next, we evaluate how the performance varies with *lesser* samples for marking (using ResNet50): we consider 63 and 32 samples by each user, which amount to $\sim 0.005\%$ and $\sim 0.0025\%$ of the training set. The results are in Fig. 10.

MembershipTracker can still maintain 0% FPR@100% TPR, when the marking ratio is reduced from 0.01% to 0.005%. However, when the marking ratio is further reduced to 0.0025%, *MembershipTracker* experiences lower auditing performance - this trend is similar to that in our ablation study in Appendix C.1.1 and thus we defer the detailed discussion to Appendix C.1.1 (due to space constraints). Nevertheless, even under such a challenging case, *MembershipTracker* still achieves reasonably good performance as shown in the blue line of Fig. 10; we leave the improvements to future study.

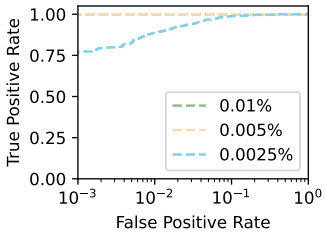


Figure 10: *MembershipTracker* under full-sized ImageNet training with different portions of samples for data marking. Aggressively reducing the portion of samples marked by each user can lead to reduced performance, though *MembershipTracker* can still maintain reasonably high auditing performance even when each user can only mark merely 0.0025% of the training data (the blue line).

7 Limitations

There are three main limitations by *MembershipTracker*, and we discuss them next.

(1) In order to trace data provenance in a given model, we assume the model is trained on a small number of target samples that are marked with *MembershipTracker*. This is important in ensuring that the target samples can be strongly memorized by the model, and the set-based MI process can leverage the collective information across the target samples to generate a reliable outcome.

However, this assumption may be invalid if: (a) the users have limited samples available for data marking; or (b) they have marked several samples, but only a subset of them are used for training the model. We discuss both cases next.

(a) The first case is dependent on the application scenario and may be overcome by the users themselves. E.g., if the users want to know if their images posted on social media platform may be misused, they can upload more images that are marked with *MembershipTracker* (both Section 6 and Appendix C.1.1 study how different number of samples marked by the users may affect *MembershipTracker*'s performance).

With that said, marking more samples does not mean all of them would be used for training the model, which leads to the second threat to the validity of *MembershipTracker*.

(b) Recall the set-based MI process is underpinned by the observation that the average loss of the samples by the non-target users are higher than that of the target users (Section 4.4.2). This, however, might not hold, if only a *subset* of the marked samples from each target user are used for model

Table 5: Evaluation on the partial inclusion of the marked samples in model training. With lesser samples (50 samples per user) included for training, the average loss on the entire set of marked samples by each target user increases, which in turn increases the FPR by *MembershipTracker*.

% of marked samples used for training	FPR@100% TPR (100 target users)	FPR@100% TPR (5 target users)
100% (50/50)	0.00	0.00
90% (45/50)	0.02	0.00
70% (35/50)	0.02	0.04
50% (25/50)	0.28	0.34
30% (15/50)	20.12	2.46
20% (10/50)	76.34	6.82

training. This is because those samples that are not used for training would yield higher prediction loss, which in turn increases the average loss on the entire set of marked samples, and affects *MembershipTracker*.

Evaluation. To understand this, we perform an experiment where we assume the model owner collects 50 samples from each user (0.2% of the training set), and he/she deliberately includes only a subset of samples from each target user for training. Table 5 shows the results, where we observe a similar trend when *MembershipTracker* is applied to support 100 target users (a more challenging setting) or 5 target users only. In both cases, the FPR incurred by *MembershipTracker* grows as the number of marked samples used for training decreases, which is due to the higher average loss on the target users' samples as we explained earlier.

Overall, we find that *MembershipTracker* can still maintain $< 0.5\%$ FPR@100% TPR, even when only 50% of the marked samples by each target user are used for training. But its performance drops rapidly when lesser samples are included for training. Nevertheless, we remark that using only a subset of the user data for training merely represents a special usage case for understanding the potential threat to *MembershipTracker*'s performance, and it does *not* necessarily represent the real-world practice of model training.

Specifically, given that DL models are most effective when applied to large datasets [5, 60], it is often in the model creator's interest to use *more data* (e.g., use all the collected data) for training, and *MembershipTracker* can achieve superior performance in such a setting. As the first of its type, how *MembershipTracker* may reshape the current model training practice remains to be seen.

(2) Secondly, despite our best effort in keeping the targeted changes from severely distorting the marked data, the artifacts created by *MembershipTracker* are still visible under close inspection (more visual examples are shown in Appendix A). However, performing manual inspection to filter out the specially-marked samples by *MembershipTracker* can still be a challenge, particularly under the large-scale training setting (e.g., with millions of images). With that said, creating more subtle targeted changes still remains an important venue for future investigation.

(3) Finally, since *MembershipTracker* targets the model's propensity in memorizing data, in principle, *any* effective measures that can mitigate the model's memorization on training data would be able to degrade the auditing performance by *MembershipTracker*. Indeed, our evaluation in Section 5.2 shows that common approaches such as differential privacy and data augmentation can be configured to degrade the auditing success by *MembershipTracker*, but they also result in utility loss to the model (undesirable).

Given that building high-performance models often represents the model creators' foremost interest, the development of tools like *MembershipTracker* can prompt the practitioners to reconsider the choice of resorting to other legal data acquisition means and building high-performance models using the authorized data.

8 Discussion and Future Work

This work presents *MembershipTracker*, a practical data provenance tool that can support ordinary users to audit whether their specific data are used to train deep learning models without their permission. If their data are found to be misused, the users can take legal action or request the "right to be forgotten" in accordance with privacy regulations such as GDPR [2] (e.g., based on techniques on machine unlearning [7]). Finally, we discuss two other future work directions.

First, while *MembershipTracker* retrofits the conventional membership inference (attack) to enable responsible AI, it also faces the risk of being misused by the malicious party. Notably, the data marked with *MembershipTracker* would now become easier to be de-identified by *any* party with access to them. However, *MembershipTracker* should be applied only when the users do *not* want their data to be used for training DL models, in which membership privacy may not be of the users' concern. By contrast, if the users are willingly sharing their data for model training, they should refrain from using *MembershipTracker* to avoid unnecessary membership exposure. Future studies can explore potential usage cases where *MembershipTracker* is applicable *and* protecting the membership privacy of the marked samples is necessary, and consider the further adaptation of *MembershipTracker* to cater to such a need.

Next, future work can explore the potential of extending *MembershipTracker* to other domains like generative models. Prior work show that performing MI on generative image models (e.g., Stable Diffusion) is also related to the model's propensity in memorizing individual data points [9]. Thus, while *MembershipTracker* demonstrates promising performance under the million-scale training setting, future study can investigate whether the large generative models would also be prone to memorize the samples marked with *MembershipTracker* under the billion-scale training environment.

References

[1] Celeba hq face identity and attributes recognition using pytorch. https://github.com/ndb796/CelebA-HQ-Face-Identity-and-Attributes-Recognition-PyTorch/tree/main.

[2] General data protection regulation. https://gdpr-info.eu.

[3] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016.

[4] Michael Aerni, Jie Zhang, and Florian Tramèr. Evaluations of machine learning privacy defenses are misleading. *arXiv preprint arXiv:2404.17399*, 2024.

[5] Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312, 2022.

[6] Martin Bertran, Shuai Tang, Aaron Roth, Michael Kearns, Jamie H Morgenstern, and Steven Z Wu. Scalable membership inference attacks via quantile regression. *Advances in Neural Information Processing Systems*, 2024.

[7] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *IEEE Symposium on Security and Privacy (SP)*, 2021.

[8] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

[9] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium*, 2023.

[10] Harsh Chaudhari, Giorgio Severi, Alina Oprea, and Jonathan Ullman. Chameleon: Increasing label-only membership leakage with adaptive poisoning. *arXiv preprint arXiv:2310.03838*, 2023.

[11] Guangke Chen, Yedi Zhang, and Fu Song. Slmia-sr: Speaker-level membership inference attacks against speaker recognition systems. *arXiv preprint arXiv:2309.07983*, 2023.

[12] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, and Yang Zhang. Face-auditor: Data auditing in facial recognition systems. *arXiv preprint arXiv:2304.02782*, 2023.

[13] Yufei Chen, Chao Shen, Yun Shen, Cong Wang, and Yang Zhang. Amplifying membership exposure via data poisoning. *Advances in Neural Information Processing Systems*, 2022.

[14] Zitao Chen and Karthik Pattabiraman. A method to facilitate membership inference attacks in deep learning models. *arXiv preprint arXiv:2407.01919*, 2024.

[15] Zitao Chen and Karthik Pattabiraman. Overconfidence is a dangerous thing: Mitigating membership inference attacks by enforcing less confident prediction. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2024.

[16] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International Conference on Machine Learning*. PMLR, 2021.

[17] Kenneth T Co, Luis Muñoz-González, Sixte de Maupeou, and Emil C Lupu. Procedural noise adversarial examples for black-box attacks on deep convolutional networks. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019.

[18] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009.

[20] Pranav Dixit. Meet the three artists behind a landmark lawsuit against ai art generators. *BuzzFeedNews*, 2023.

[21] Josh Dzieza. https://www.theverge.com/features/23764584/ai-artificial-intelligence-data-notation-labor-scale-surge-remotasks-openai-chatbots. *The Verge*, 2023.

[22] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[24] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joseph Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. In *International Conference on Machine Learning*. PMLR, 2022.

[25] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2016.

[26] Kashmir Hill. Facial recognition's 'dirty little secret': Millions of online photos scraped without consent. *The New York Times*, 2020.

[27] Hongsheng Hu, Zoran Salcic, Gillian Dobbie, Jinjun Chen, Lichao Sun, and Xuyun Zhang. Membership inference via backdooring. In *The 31st International Joint Conference on Artificial Intelligence (IJCAI-22)*, 2022.

[28] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[29] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[30] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. *arXiv preprint arXiv:2101.04898*, 2021.

[31] Zonghao Huang, Neil Zhenqiang Gong, and Michael K Reiter. A general framework for data-use auditing of ml models. *arXiv preprint arXiv:2407.15100*, 2024.

[32] Nikhil Kandpal, Krishna Pillutla, Alina Oprea, Peter Kairouz, Christopher A Choquette-Choo, and Zheng Xu. User inference attacks on large language models. *arXiv preprint arXiv:2310.09266*, 2023.

[33] Konstantin Kirchheim, Marco Filax, and Frank Ortmeier. Multi-class hypersphere anomaly detection. In *26th International Conference on Pattern Recognition (ICPR)*, 2022.

[34] Konstantin Kirchheim, Marco Filax, and Frank Ortmeier. Pytorch-ood: A library for out-of-distribution detection based on pytorch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022.

[35] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[36] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

[37] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 2018.

[38] Guoyao Li, Shahbaz Rezaei, and Xin Liu. User-level membership inference attack against metric embedding learning. *arXiv preprint arXiv:2203.02077*, 2022.

[39] Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. *Advances in Neural Information Processing Systems*, 35:13238–13250, 2022.

[40] Yingwei Li, Song Bai, Yuyin Zhou, Cihang Xie, Zhishuai Zhang, and Alan Yuille. Learning transferable adversarial examples via ghost networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2020.

[41] Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. The artbench dataset: Benchmarking generative models with artworks. *arXiv preprint arXiv:2206.11404*, 2022.

[42] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 2020.

[43] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[44] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[45] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, 2015.

[46] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *International Conference on Learning Representations*, 2020.

[47] Matthieu Meeus, Shubham Jain, Marek Rei, and Yves-Alexandre de Montjoye. Did the neurons read your book? document-level membership inference for large language models. In *33rd USENIX Security Symposium*, 2024.

[48] Yuantian Miao, Xue Minhui, Chao Chen, Lei Pan, Jun Zhang, Benjamin Zi Hao Zhao, Dali Kaafar, and Yang Xiang. The audio auditor: user-level membership inference in internet of things voice services. *Proceedings on Privacy Enhancing Technologies*, 2021.

[49] Yannic Neuhaus, Maximilian Augustin, Valentyn Boreiko, and Matthias Hein. Spurious features everywhere-large-scale detection of harmful spurious features in imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20235–20246, 2023.

[50] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[51] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022*, 2021.

[52] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[53] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *International Conference on Machine Learning*. PMLR, 2020.

[54] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*. PMLR, 2019.

[55] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. Fawkes: Protecting privacy against unauthorized deep learning models. In *29th USENIX security symposium*, 2020.

[56] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.

[57] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, 2017.

[58] Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[59] Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *Advances in Neural Information Processing Systems*, 2022.

[60] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, 2017.

[61] Yiyou Sun and Yixuan Li. Dice: Leveraging sparsification for out-of-distribution detection. In *European Conference on Computer Vision*. Springer, 2022.

[62] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*. PMLR, 2022.

[63] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[64] Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. Mitigating membership inference attacks by self-distillation through a novel ensemble architecture. In *31st USENIX Security Symposium*, 2022.

[65] Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[66] James Vincent. Ai art tools stable diffusion and midjourney targeted with copyright lawsuit. *The Verge*, 2023.

[67] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. Vim: Out-of-distribution with virtual-logit matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[68] Emily Wenger, Xiuyu Li, Ben Y Zhao, and Vitaly Shmatikov. Data isotopes for data provenance in dnns. *Proceedings on Privacy Enhancing Technologies*, 2024.

[69] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[70] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[71] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018.

[72] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[73] Yi Zeng, Minzhou Pan, Hoang Anh Just, Lingjuan Lyu, Meikang Qiu, and Ruoxi Jia. Narcissus: A practical clean-label backdoor attack with limited information. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023.

[74] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

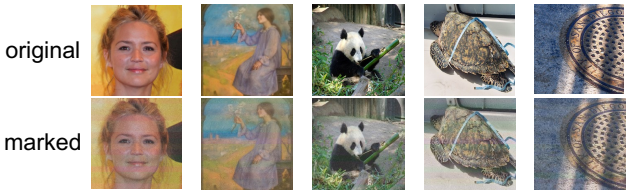


Figure 11: Visualization of the original and marked samples.

A Model Training Details

For CIFAR10 and CIFAR100, we train a WideResNet-28-4 model on 25,000 samples for 100 epochs, with a learning rate of 0.1 (decayed by 5 at epoch 60, 80, 90), a weight decay of 5e-4 and momentum of 0.9. For TinyImageNet and ArtBench, we fine-tune an ImageNet-pretrained ResNet-18 model on 25,000 samples. For CelebA, we first follow [1] to create a 307-class subset from the dataset, and then similarly fine-tune an ImageNet-pretrained model on 2,000 samples. For fine-tuning, we use a small fine-tuning rate of 0.01 with 30 epochs and a momentum of 0.9. In all datasets except CelebA (where we find that fine-tuning on the entire set yields better accuracy), we use 20% of the training data as the validation set. We also use common data augmentations such as random cropping and horizontal flip.

Table 6: Evaluation on different *models* (parenthesized numbers show accuracy diff. compared with the original models).

Model	FPR@ 100% TPR	Model accuracy
WideResNet	0.00	66.34 (-0.41)
ResNet	0.00	64.49 (-0.69)
ResNext	0.00	63.43 (-0.47)
DenseNet	0.00	67.14 (-0.16)
GoogleNet	0.00	69.09 (-0.33)
SeNet	0.00	65.26 (-0.37)

Table 7: Evaluation on different *training-set sizes*.

Training-set size	FPR@ 100% TPR	Model accuracy
25,000	0.00	66.34 (-0.41)
20,000	0.00	63.73 (-0.16)
15,000	0.00	58.99 (-0.83)
10,000	0.00	50.28 (-0.09)
5,000	0.00	35.74 (-0.10)

B Detailed Results for Section 5.1

Earlier in Section 5.1 we evaluate *MembershipTracker*'s performance under different models and training-set sizes. The detailed results for this are in Table 6 and Table 7. In both cases, *MembershipTracker* is able to achieve 0% FPR@100% TPR with $< 1\%$ accuracy drop the model.

C Ablation Study

MembershipTracker consists of a data marking and set-based MI process, and we conduct a detailed ablation study into these two in Appendix C.1 and C.2.

C.1 Two-step Data Marking

We first evaluate the effectiveness of the two data-marking components (Appendix C.1.1). Meanwhile, we also study how *MembershipTracker*'s performance varies under different number of target samples contributed by the users.

Next, we evaluate how different data-marking intensities may affect *MembershipTracker*'s performance (Appendix C.1.2). Finally, we explore alternative data marking methods in Appendix C.1.3.

C.1.1 Investigating the two-step data marking process

This section compares the performance of different variants of *MembershipTracker*, including (1) the full *Membership-*

Table 8: Ablation study of *MembershipTracker*'s two-step data marking process: (a): Image blending with OOD feature; (b): Perlin noise injection. Each *column* reports the performance under a specific number of target samples contributed by the users (5∼15 samples by each).
(1) From top to the bottom: both (a) and (b) are crucial to *MembershipTracker*'s high auditing performance.
(2) From left to the right: *MembershipTracker*'s auditing performance degrades as the number of samples contributed by the users reduces. However, even in the challenging case where each user marks only 5 samples, the full *MembershipTracker* still incurs only 0.64% FPR@100% TPR.

Approach	FPR@100% TPR		
	15 samples (0.06% dataset)	10 samples (0.04% dataset)	5 samples (0.02% dataset)
(a) + (b)	0.00%	0.18%	0.64%
(a)	0.10%	0.96%	4.62%
(b)	0.06%	0.40%	1.8%
None	0.34%	1.54%	7.18%

Tracker with image blending and noise injection; variant (2) with image blending only; variant (3) with noise injection only; and variant (4) without any data marking.

We consider a total of 100 target users, and each user contributes different number of samples (5 to 15) to the training set (25,000 instances). Table 8 reports the results, and there are two main findings.

(1) **The two-step data marking process is vital for *MembershipTracker*'s high auditing success**. The full technique with the two-step process consistently achieves lower FPR than the two other variants with only a single-step marking (second and third row in Table 8). The last variant (the bottom row in Table 8) considers the set-based MI process on the original target samples without any data marking, which suffers from even higher FPR. E.g., under 5 target samples per user, the full technique has only 0.64% FPR vs. 7.18% by the one without data marking.

Meanwhile, the performance difference between different approaches in Table 8 shrinks as the number of target samples contributed by the users increases. Thus, there is a low FPR even when auditing the original target samples (without data marking), e.g., 0.34% FPR@100% TPR on the setting with 15 target samples per user. This is in fact due to the effectiveness of the proposed set-based MI process (without which, performing the standard instance-based MI incurs 44% FPR@100% TPR), and the ablation study on the set-based MI process is in Appendix C.2.

(2) **Performance variation under different number of target samples by the users**. Table 8 also shows *MembershipTracker*'s performance degrades as the number of target samples by the users reduces (this trend is also similar to our earlier evaluation of the ImageNet training experiment in Section 6). There are two reasons.

First, recall that in *MembershipTracker*, each user applies a random outlier feature to mark their data, and with lesser data for marking, the outlier feature appears in fewer samples. Thus, the model's memorization on the marked samples (that contain the outlier feature) subsides. This can be reflected from the higher prediction loss on the samples, and we validated this.

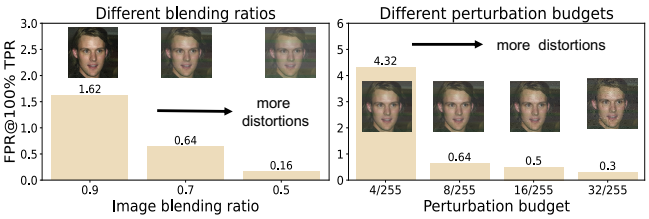


Figure 12: Performance evaluation under different image blending ratios (*left*) and noise perturbation budgets (*right*) (5 target samples per user). Zoom in to view the samples marked with different intensities.

Secondly, the limited size of target samples also reduces the amount of information available to the set-based MI process, which can result in a lower auditing performance as well. With that said, even in the challenging case where each user can only mark 5 samples (0.02% of the dataset), the full *MembershipTracker* still incurs a reasonably low FPR of 0.64% with 100% TPR.

C.1.2 Data marking with different parameters

There are two parameters: the blending ratio m and the noise perturbation budget σ. Our main experiments use $m = 0.7$ and $\sigma = 8/255$, and we now study how different parameters may affect *MembershipTracker*'s performance. The results are in Fig. 12, where we consider a more challenging case with 5 target samples per user, because the performance difference between different marking intensities is less distinctive when there are more target samples by each user (e.g., 15).

In both figures of Fig. 12, higher marking intensity indeed enhances the ability of the marked samples to be memorized by the model, and contributes to the better performance by *MembershipTracker*. This illustrates that the users can moderate the balance between the amount of distortion applied to their data and the degree of auditing performance obtained from the resulting data.

C.1.3 Alternative data marking methods

(1) Alternative outlier feature generation. In addition to constructing outlier features as samples consisting of random color stripes, we consider alternative strategies by using samples from an OOD dataset. Specifically, we use random samples from the TinyImageNet and CelebA dataset, which are blended into the CIFAR100 samples. The results are shown on the left of Fig. 13, and a visualization on the right.

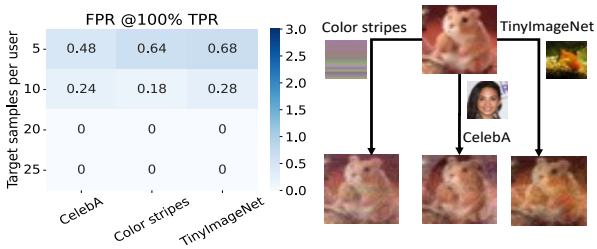


Figure 13: *MembershipTracker* maintains similarly high performance when using different types of outlier features for data marking, such as samples with random color stripes, samples from an OOD dataset (TinyImageNet and CelebA).

Table 9: Comparing the instance-based and the proposed set-based MI process. The latter consistently outperforms the former method by leveraging the average loss across the target samples by each user to perform MI auditing.

Approach	FPR@100% TPR		
	10 samples (0.04% data)	15 samples (0.06% data)	25 samples (0.10% data)
Instance-based MI	36.09	20.47	7.97
Set-based MI	0.18	0.00	0.00

As shown, the resulting samples can be similarly memorized by the model and enable *MembershipTracker* to maintain high provenance success. E.g., in the last two rows on the left of Fig. 13 (where each user marks 20 or 25 samples), using different types of outlier features can similarly avail the users to achieve 0% FPR@100% TPR. This offers the flexibility for the users to choose different types of outlier features to mark and protect their data.

(2) Alternative noise injection. We compare injecting perlin noise vs. uniform random noise. We find that perlin noise yields better results (details omitted), because perlin noise is a stronger adversarial noise [17], and thus more capable of inducing the model to memorize the marked samples. Other alternatives such as using other noises like gabor noise [17] may also be explored in future studies.

C.2 Set-based MI Verification

This section compares the set-based MI with the common instance-based MI process. Table 9 shows the results where the target samples are marked with *MembershipTracker* (and we observe a similar trend when comparing both approaches on the original samples without data marking). As in Table 9, the proposed set-based MI process consistently outperforms the instance-based MI method. E.g., when each target user can mark only 15 samples (the second column in Table 9), they can achieve 0% FPR@100% TPR via the set-based MI process, while the instance-based MI incurs 20.47% FPR.

Meanwhile, the FPR by the instance-based MI approach reduces as the number of marked samples contributed by the users increases (this trend is similar to that in Table 8 discussed earlier). Nevertheless, even if each user can mark 25 samples, the instance-based MI method still incurs 7.97% FPR@100% TPR, while the set-based MI has 0% FPR.

D Comparison with Wenger et al. [68]

Wenger et al. propose a technique to audit the unauthorized use of data in training DL models. It first injects a target spurious feature into the user data, and then detects whether the model has associated the injected feature with a target class label. To detect, the users can separately overlay the target feature and some non-target features to an auxiliary set of samples (outside the target class), and then check whether the target feature causes a slightly higher probability shift to the target class in those samples, compared with the non-target features. For instance, assume the target class is "cat". The user may compare the cat class probability on some "dog" images that are overlaid with the target and non-target features (e.g., 10% vs. 1%), and detect potential data misuse.

In the following, we first discuss a notable issue we found in their work, which leads to a significant underestimate of false positives by their technique; and then further compare both techniques.

Understanding the problematic evaluation in Wenger et al [68]. In their work, a testing feature should be detected as being used for training, if it causes higher probability shift than the *non-target* feature - this procedure should be the **same** regardless of whether the testing feature has actually been used for training or not (i.e., true or false positive). Unfortunately, this is not the case in [68].

We use Alg. 2 as an example to explain in the following. Assume T_{in} is the true target feature that causes a probability shift of 10%, and T_{out} 5% and T'_{out} 1% (T_{out} and T'_{out} are two random non-target features that are *not* used for training). Line 1 in Alg. 2 compares T_{in} with T'_{out} to check true positive.

To compute false positives (Line 2), however, Wenger et al. [68] simply *invert* the order of Line 1, which essentially compares the probability shift between the non-target feature T'_{out} and the *true target* feature (T_{in}). This would report *no* false positives (because T'_{out} has a lower probability shift than T_{in}), but it is not the correct way to compute false positive.

Instead, a fair evaluation should: (1) sample a *new non-target* feature (T_{out}); and (2) compare its probability shift relative to *another non-target* feature (T'_{out})[3]. This is illustrated in Line 3 of Alg. 2 (which is basically the same as Line 1 for detecting true positive). In such a rectified procedure, the technique *would have* a false positive, because T_{out} has a higher probability shift than T'_{out}.

[3] In fact, the order of T_{out} and T'_{out} in Line 3 of Alg. 2 can be interchangeable, as they are *both* random features that are not used for training. We use the given order only to illustrate how Wenger et al. [68] could overlook a false positive in their original detection.

Algorithm 2 Illustrating the problematic evaluation in measuring *false positive* by [68], and how to fix it.

Input: T_{in}: the *true target* spurious feature used for model training;
T_{out}, T'_{out}: two random *non-target* features (*not* used for training);
/* Assume the prob shift by $T_{in}, T_{out}, T'_{out}$ to be 10%, 5% and 1% */

/* ✓ Compare T_{in} with T'_{out} to check (true) positive */
1: **if** Prob shift by $T_{in} > T_{out}{}'$ **then** true positive += 1 **end if**

/* ✗ [68] invert the order of Line 1 to check (false) positive */
2: **if** Prob shift by $T'_{out} > T_{in}$ **then** false positive += 1 **end if**
/* This would report *no* false positive */

/* ✓ Sample a new T_{out} and compare it with T'_{out} for FP */
3: **if** Prob shift by $T_{out} > T'_{out}$ **then** false positive += 1 **end if**
/* This would report a false positive */
/* ✗ But it still cannot control the test under a specific low FPR */

/* ✓ A principled way to satisfy the above requirement */
4: Estimate the prob shift by sampling a large number of non-target features
/* α can be a small number such as 0.01% (for the low FPR) */
5: $c \leftarrow (1-\alpha)$-percentile of the CDF for the above estimated prob shift
6: **if** Prob shift by $T_{test} > c$ **then**
7: **if** T_{test} belongs to T_{in} **then** true positive += 1
8: **else** false positive += 1
9: **end if**
10: **end if**

Validation. To evaluate how this affects their results, we use the implementation from the authors [68]. We use CIFAR100 dataset with 50 target classes and 50 non-target features for comparison; and otherwise follow the the same parameters they used in their experiment [68]. We first validate that we are able to obtain similar results to their reported performance, with 99.63% TPR and 0% FPR[4].

To re-evaluate the FPR following Line 3 in Alg. 2, we compare the probability shift between two disjoint sets of non-target features (T_{out}, T'_{out}). Because both feature sets are not used for training, the FPR should be similar if we compare T_{out} Vs. T'_{out} or T'_{out} Vs. T_{out} (i.e., their order is interchangeable). We indeed did observe this (with 37% and 34.6% FPR). On average, their technique incurs a 35.8% FPR.

Wenger et al. [68] do mention that comparing two sets of non-target features is equivalent to random guessing, and we also observe the same. However, this "indistinguishablity" applies only when attempting to distinguish two *set* of features; if we consider two random *individual* features, it is likely that one feature will cause a higher/lower probability shift than the other. This is the exact issue with Wenger et al.'s method, which performs each test on a *feature-to-feature* level (Line 3 in Alg. 2). Thus, it still permits a large number of FPs.

How to reduce (and control) FPR? One way to reduce the FPR in [68] is to employ a tighter significance level λ, which is used to detect whether the probability shift caused by the target feature is statistically higher than the non-target feature, and it defaults to be 0.1 in [68]. However, this will also degrade the TPR, e.g., when we change to use a λ of 1e-8, the FPR is reduced to 1.9%[5], and the TPR is already degraded to 86.4%[6]. In addition, while using a tighter significance level can reduce the FPR, it still *cannot* control the FPR within a specific low regime (e.g., 0.01%).

To fulfill this requirement, a principled way is to adapt a similar setup established in existing MI studies [6, 8, 70]. This procedure is sketched in Line 4 to Line 10 in Alg. 2. However, based on our results in the previous paragraph, it is likely that the TPR by their technique would further decrease when the TPR is controlled at a lower FPR regime (e.g., 0.01%).

Further comparison with [68]. We further compare Wenger et al. with *MembershipTracker* in our setting, where we evaluate how effective is their approach, if adapted for MI-based data provenance.

Specifically, we consider 100 target users and each user marks 25 samples (0.1% of the training set). As in Wenger et al. [68], we use ImageNet images as the outlier features to be blended into the CIFAR100 samples, and *MembershipTracker* uses random color stripes as before. [68] have a configurable parameter to control the intensity of the outlier feature when overlaid to the target data - we thus compare *MembershipTracker* with their approach under *varying* marking intensities.

Fig. 14 shows the results, and Fig. 15. shows a visualization of the samples marked with different intensities.

As shown in Fig. 14, compared with the baseline approach without any data marking, the method by Wenger et al. can indeed increase the MI success. For instance, the green line in Fig. 14 (marked using $m = 0.6$, which is equivalent to the

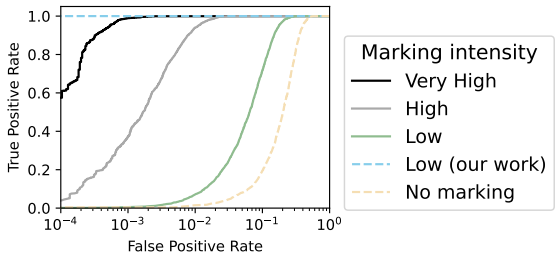


Figure 14: Comparing the technique by Wenger et al. (under varying data marking intensities) and *MembershipTracker* in terms of their MI effectiveness for tracing data provenance. Increasing the marking intensity in their approach can improve the MI success, but it: (1) still has *significantly lower* MI success than *MembershipTracker*; and (2) already incurs *considerably higher* distortion to the images than *MembershipTracker* (visualization in Fig. 15).

[4] We identified a bug in their code (an issue related to pass-by-reference in Python that results in the code using a different set of auxiliary data for marking the target and non-target features), and we fixed it in our evaluation. We have also notified the authors about this bug in their official code repository using an anonymous account.

[5] We also checked that larger λ like 1e-10 cannot further reduce FPR.

[6] Another option to reduce FPR is to adjust the proportion of positive tests (δ) for the tool to return a positive outcome (details omitted): we increased δ from 0.6 (their default setting) to 1.0, and still observed a similar trade-off.

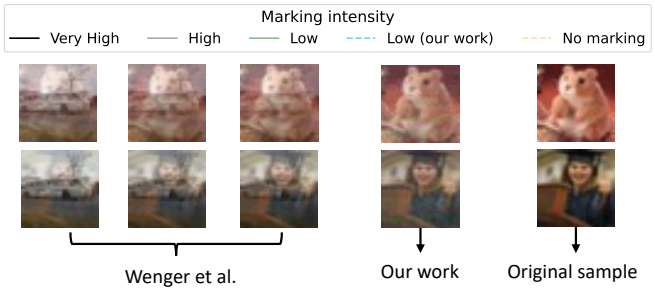


Figure 15: Samples marked with different intensities by the method of Wenger et al. vs. the samples marked by *MembershipTracker*. Our approach incurs significantly lower distortion to images than Wenger et al., and still outperforms their approach with much higher MI success.

default marking intensity in [68]) increases the AUC from 0.7927 to 0.9253. However, this still has very limited performance in the low FPR regime. Increasing the marking intensity in their approach can further improve the MI success, but it: 1) comes at the cost of higher image distortion; and 2) still has lower auditing performance than *MembershipTracker*.

For instance, under $m = 0.5$ (meaning 50% of the original features are replaced with the outlier features), their approach still yields much lower MI success than *MembershipTracker* (the gray Vs. blue line in Fig. 15), and the samples marked by their approach already become highly conspicuous compared with the original samples (the second and fifth column in Fig. 15). Using higher marking intensity ($m = 0.3$) still suffers from a similar trade-off.

Our earlier discussion in Section 4.3.2 explains that injecting outlier features alone is not enough, as the original features hinder the model's memorization on the outlier features. To overcome this, we propose a novel approach to inject a small amount of *perlin noise*, which works similar to adversarial perturbation and can significantly enhance the model's memorization on the marked samples (as evaluated earlier in Appendix C.1.1) while preserving high image quality (the fourth column in Fig. 15).

Combined with the proposed set-based MI process (another key component in our work), *MembershipTracker* is able to achieve perfect MI success (the blue line in Fig. 14) and outperforms the method by Wenger et al.

Summary. While both Wenger et al. and our work aim at detecting the unauthorized use of personal data in training DL models, their original technique greatly underestimates the detection false positives, and thus still struggles with achieving high TPR and low FPR.

We also realign their technique for MI-based provenance. We find *MembershipTracker* outperforms their approach by: 1) incurring lower distortion to the data; and 2) achieving significantly higher auditing performance. This renders *MembershipTracker* a more practical data provenance tool.

Table 10: Comparing the technique by Tramer et al. and *MembershipTracker* in terms of their MI effectiveness for tracing data provenance.
Their approach works by injecting mislabeled samples to increase the MI success, which can reduce the FPR, but only to a limited extent.
In comparison, *MembershipTracker* does not require mislabeling any data, and still achieves considerably lower FPR.

Approach	FPR@100% TPR		
	5 samples (0.02% data)	15 samples (0.06% data)	25 samples (0.1% data)
Original	49.33	54.79	56.32
Tramer et al.	10.80	6.96	4.92
MembershipTracker	0.64	0.00	0.00

E Comparison with Tramer et al. [65]

This section compare *MembershipTracker* with another related work (Tramer et al). Their work proposes a technique to improve the MI success based on data poisoning. The main idea is to inject mislabeled samples into the model's training set, which can transform the target samples into outliers and amplify their influence to the model's decision, thereby making the target samples easier to be de-identified.

Tramer et al. is the closest work to ours that can improve the MI success and without requiring the expensive shadow-model calibration. However, it assumes the users can mislabel their data, which can be challenging in the real-world scenarios where the users do not have control over the data labels [19, 21, 68]. In comparison, *MembershipTracker* makes no such assumption and it only requires the users to mark their data with some small and targeted changes, which is a more realistic setting.

For completeness, we evaluate how effective their approach is, *if* the users are able to mislabel their data for provenance purpose. We follow the targeted label flipping attack in Tramer et al., where for each target instance (x, y), we inject multiple mislabeled data points $D_{\text{adv}} = \{(x, y'), ..., (x, y')\}$ for some label $y' \neq y$ [65]. We inject different number of poisoned samples and compare their approach with *MembershipTracker*.

Table 10 shows the results, where the method by Tramer et al. indeed increases the exposure against the targeted samples, and the FPR@100% TPR is reduced from 49.33%∼56.32% to 4.92%∼10.80%. Fig. 16 shows the ROC curve under the setting of 15 samples per user (similar trend on other settings). Meanwhile, the second row in Table 10 also shows that injecting more poisoned samples can contribute to higher MI success, which is also in line with [65]. Nevertheless, even in the case where each user contributes 25 samples, the method by Tramer et al. still incurs 4.92% FPR@100% TPR; while *MembershipTracker* has 0% FPR@100% TPR.

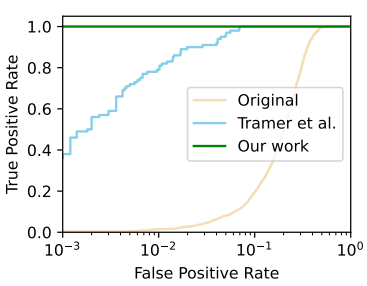


Figure 16: Comparing the performance between the approach by Tramer et al. and *MembershipTracker* (under 15 samples per user), and *MembershipTracker* achieves much higher MI success than their approach.

F Evaluating Huang et al. [31] under Limited Data Setting

Concurrent work by Huang et al. introduces a general framework for auditing unauthorized data use in training DL models [31]. Their main idea is to generate two perturbed copies of the target data, randomly publish one of them, and then compare the model's membership score on the published Vs. the unpublished one.

While both their work and *MembershipTracker* have a similar goal, their work assumes the users can mark a large portion of data (1%∼10%). However, in a real-world setting, model creators often curate their dataset by scraping data from multiple sources, and thus the data collected from each data holder may only constitute a small proportion of the dataset. For this, *MembershipTracker* targets a more realistic (and challenging) scenario, where the users can modify only a limited amount of data ($\leq 0.1\%$).

To understand the performance of their technique under such a limited-data setting, we use the original implementation from the authors [31] for an evaluation. Their approach can be applied to both image classifiers and foundation models, and we evaluate their approach under the same classifier setting as *MembershipTracker* (using CIFAR100). We first consider marking 1% of the dataset, and find that their technique can still maintain 100% detection success rate.

We then *stress test* their technique using 0.2% marking percentage, under which their auditing performance degrades considerably: the detection success rate is reduced to 65% when the output vector is available. Under the label-only setting, the success rate further drops to 25%. Therefore, our evaluation indicates that auditing data use under the limited-data setting still remains a major challenge, and *MembershipTracker* represents a desirable technique that can offer superior auditing performance in these settings.

With that said, their framework can still be used for other application domains that *MembershipTracker* currently does not support, such as foundation models. In those applications, their tool remains the state-of-the-art data auditing solution, and we leave the extension of *MembershipTracker* to other domains to future work.

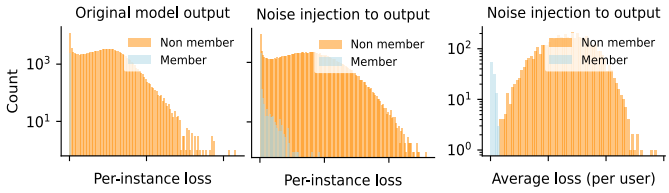


Figure 17: Output noise injection increases the prediction loss of the member samples (from the leftmost to the middle figure), but their loss are still lower than many non-member samples'. Thus, *MembershipTracker* can leverage the average loss of the samples by each user to achieve a 0% FPR@ 100% TPR (the rightmost figure).

G Detailed Results for Output Perturbation Defense

Section 5.2.7 studies two defense strategies to perturb the model's outputs: (1) injecting Gaussian noise to the prediction outputs; and (2) returning only the top-1 prediction label. We discuss the detailed results below.

(1) For the first method, we find that injecting Gaussian noise (using zero-mean noise with standard deviation σ in 0.5∼5) to the output vectors is not an effective solution, and *MembershipTracker* can still maintain very low FPR even if the defense has caused severe accuracy degradation. E.g., injecting Gaussian noise with $\sigma = 3$ degrades the accuracy from 65.32% to 52.05%, under which *MembershipTracker* still achieves a 0% FPR@100% TPR. We use Fig. 17 to explain next.

As in the first two figures of Fig. 17, injecting noise to the outputs increases the prediction loss on the member samples (and non-members too). Hence, more member samples (the blue area in the middle figure of Fig. 17) now have similar loss as the non-member samples (undesirable).

Fortunately, *MembershipTracker* can still overcome this via its set-based MI process. This is because, the prediction loss of the member samples, though increased, are still much lower than that of many non-member samples. Thus, there is still a clear difference between the average loss for the member and non-member users' samples (the rightmost figure in Fig. 17), from which *MembershipTracker* is able to achieve a 0% FPR@100% TPR.

(2) In the **label-only** setting, since the class probabilities are not available, we use a simple method to compute a proxy for the prediction loss: we assign a loss value of 0 for the correctly-classified samples and 1 for other samples. In this case, all the correctly-classified non-members have the same 0 loss as the member samples. However, we find that *MembershipTracker* can still overcome the defense and achieve a 0.04% FPR@100% TPR.

The reason is that there is a major accuracy difference between the member and non-member users' samples: 100% vs. 22.02% (the latter yield low accuracy as they are marked with the random outlier features and perlin noise, but they are *not* present in the training set). Thus, the "average loss" for the samples by the member users are still lower than that by the non-member users. This enables *MembershipTracker* to succeed even under the label-only setting.