A General Framework for Data-Use Auditing of ML Models

Zonghao Huang
Duke University
Durham, NC, USA
zonghao.huang@duke.edu

Neil Zhenqiang Gong
Duke University
Durham, NC, USA
neil.gong@duke.edu

Michael K. Reiter
Duke University
Durham, NC, USA
michael.reiter@duke.edu

Abstract

Auditing the use of data in training machine-learning (ML) models is an increasingly pressing challenge, as myriad ML practitioners routinely leverage the effort of content creators to train models without their permission. In this paper, we propose a general method to audit an ML model for the use of a data-owner's data in training, without prior knowledge of the ML task for which the data might be used. Our method leverages any existing black-box membership inference method, together with a sequential hypothesis test of our own design, to detect data use with a quantifiable, tunable false-detection rate. We show the effectiveness of our proposed framework by applying it to audit data use in two types of ML models, namely image classifiers and foundation models.

CCS Concepts

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

Keywords

Data-use auditing, data tracing, membership inference

ACM Reference Format:
Zonghao Huang, Neil Zhenqiang Gong, and Michael K. Reiter. 2024. A General Framework for Data-Use Auditing of ML Models. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690226

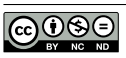
1 Introduction

The advances of machine learning (ML) models hinge on the availability of massive amounts of training data [16, 26, 32, 33, 38, 46, 82]. For example, Contrastive Language-Image Pre-training (CLIP), developed by OpenAI, is pretrained on 400 million of pairs of images and texts collected from the Internet [52], and large language models like Llama 2, developed by Meta AI, are pretrained and fine-tuned on trillions of tokens [70]. Although the development of these large ML models has significantly contributed to the evolution of artificial intelligence, their developers often do not disclose the origins of their training data. This lack of transparency raises questions and concerns about whether appropriate authorization to use this data to train models was obtained from their owners. At the same time, recent data-protection regulations, such as the General Data Protection Regulation (GDPR) in Europe [44], the California Consumer Privacy Act in the US [1], and PIPEDA privacy legislation in Canada [14], grant data owners the right to know how their data is used. Therefore, auditing the use of data in ML models emerges as an urgent and important problem.

Data auditing refers to methods by which data owners can verify whether their data was used to train an ML model. Existing methods include *passive* data auditing and *proactive* data auditing. Passive data auditing, commonly referred as membership inference [7, 13, 27, 60, 78], infers if a data sample is a member of an ML model's training set. However, such passive techniques have an inherent limitation: they do not provide any quantitative guarantee for the false-detection of their inference results. In contrast, proactive data auditing techniques embed marks into data before its publication [24, 35, 36, 55, 69, 74, 77] and can provide detection results with false-detection guarantees [55]. The existing proactive data auditing methods mainly focus on *dataset auditing* [24, 35, 36, 55, 69], where the whole training set of the ML model is contributed from one data owner and thus the data owner has control over the whole dataset, including, e.g., knowledge of the labels [35, 36, 55]. This limits their application in a real-world setting where the training dataset might be collected from multiple data owners or data sources. In addition, the existing works focus on a particular type of ML model, e.g., image classifiers [35, 36, 55, 77], and do not directly generalize to other domains. Therefore, there is a need to design a general proactive data auditing framework that requires no assumption on the dataset curation (e.g., data labeling) and can be applied to effectively audit data across various domains.

In this work, we propose such a proactive data-auditing framework. In a nutshell, the contribution of this framework is to turn any passive membership-inference technique into a proactive data-auditing technique with a quantifiable and tunable false-detection rate (i.e., probability of falsely detecting data use in an ML model). Our framework consists of a data marking algorithm and a detection algorithm. The data marking algorithm, which the data owner applies prior to data publication, generates *two* versions of each raw datum; each version is engineered to preserve the utility of the raw datum from which it is generated, but otherwise the versions are perturbed with maximally different marks. Taking the example of an image, the marks are pixel additions to the raw image that preserve its visual quality but maximize the difference between the two marked versions. Critically, this marking step is agnostic to the ML task (including, e.g., labels) in which the data versions might be used. The data owner then publishes only *one* of the two versions, chosen uniformly at random, and keeps the other hidden.

The key insight of our framework is that once the model is accessible (even in only a black-box way), any "useful" membership-inference technique should more strongly indicate the use of the published version than of the unpublished version, if the model was trained using the data-owner's published data. If the model was *not* trained using the data-owner's published data, then the



CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690226

membership-inference technique might indicate either the published or unpublished version as more likely to have been included. However, because the published version was chosen uniformly, the version that the membership test more strongly indicates was used should be equally distributed between the two.

This insight enables us to design a sequential hypothesis test of the null hypothesis that the ML model was *not* trained using the data-owner's data. Using any membership-inference test, the data owner queries the model on both the published and unpublished versions of each datum (possibly obscured to avoid detection, as we will discuss in Sec. 5.3.3), keeping a count of the times the published version was reported as having been used with greater likelihood. We derive a test to determine when the data owner can stop and reject the null hypothesis, concluding that the model was trained with her published data, *with any desired false-detection rate*.

We study the performance of our proposed framework in two cases: image classifiers and foundation models. An image classifier is a type of ML model used to assign labels to images based on their content [16, 25, 61], while foundation models are general-purpose, large ML models [6, 18, 52, 70]. In the first case, our results on multiple visual benchmark datasets demonstrate that our proposed framework effectively audits the use of the data-owner's data in image classifiers across various settings. Moreover, our proposed method outperforms the existing state-of-the-art data auditing methods, notably Radioactive Data [55] and Untargeted Backdoor Watermark-Clean (UBW-C) [35]. We also investigate adaptive attacks that the ML practitioner might use to defeat our auditing method. While our results show that certain adaptive attacks like early stopping and differential privacy can degrade the detection performance of our method, they do so at the cost of significantly diminishing the utility of the model. For the case of foundation models, we extended our evaluation to three types of foundation models: a visual encoder trained by self-supervised learning [12], Llama 2 [70], and CLIP [52]. Our results show that the proposed data auditing framework achieves highly effective performance across all of these foundation models. Overall, our proposed framework demonstrates high effectiveness and strong generalizability across different types of ML models and settings.

To summarize, our contributions are as follows:

- We propose a novel and general framework for proactive data auditing. Our framework has a simple data-marking algorithm that is agnostic to any data labeling or ML task, and a novel detection algorithm that is built upon contrastive membership inference and a sequential hypothesis test that offers a tunable and quantifiable false-detection rate.
- We demonstrate the effectiveness of the proposed framework by applying it to audit the use of data in two types of ML models, namely image classifiers and foundation models, under various settings.

Due to space constraints, some of our results are detailed only in the full paper [28]. A source code implementation of our framework is available at https://github.com/zonghaohuang007/ML_data_auditing.

2 Related Work

2.1 Data Auditing

Data auditing is a type of *proactive* technique that a data owner can use to audit the use of her data in a target ML model [24, 35, 36, 55, 69, 74]. Such methods usually include a marking algorithm that embeds marks into data, and a detection algorithm that tests for the use of that data in training a model. Radioactive Data [55] is a state-of-the-art method for auditing an image classifier, which we consider as one of our baselines in Sec. 5.2. In the marking step, Radioactive Data randomly samples *class-specific* marks and embeds them into a subset of the training dataset. In the detection step, it detects if the parameters of the final layer of the target image classifier are correlated with the selected marks, by a hypothesis test whose returned p-value is its false-detection rate. However, Radioactive Data assumes that the data owner has full control over the training set, including that, e.g., she knows the labels of the dataset and can train a surrogate model (i.e., a model similar to the target model) used to craft marked images. In contrast, our work relaxes the requirement for one data owner to control the entire training dataset. Another marking-based technique that, like ours, relaxes this requirement for classifiers is that of Wenger, et al. [77]. However, unlike ours, this technique requires most of marked data contributed by a data owner to be assigned the same label by the ML practitioner; does not provide a rigorous guarantee on the false-detection rate; and to achieve good detection performance in their reported experiments on image classifiers, needed marks that were sufficiently visible to diminish image quality.

Another line of works on image dataset auditing [35, 36, 69] is based on backdoor attacks [23, 57] or other methods (e.g., [24]) to enable a data-owner to modify her data and then detect its use to train an ML model by eliciting predictable classification results from the model (e.g., predictable misclassifications of poisoned images for backdoor-based methods). Their detection algorithms are also formulated by a hypothesis test, but they do not provide rigorous guarantees on their false-detection rates. Moreover, these methods again require the data owner's full control over the training set, in contrast to our method. In Sec. 5.2, we consider one backdoor-based auditing method, namely Untargeted Backdoor Watermark-Clean (UBW-C) [35], as one of our baselines.

To our knowledge, all existing data auditing methods focus on a particular type of ML model, e.g., image classifiers [24, 35, 36, 55, 69, 77], language models [76], or text-to-image diffusion models [74]. So, their proposed techniques do not directly generalize to other domains. In contrast, the marking algorithm in our proposed framework does not rely on any prior knowledge of the ML task (e.g., labels assigned by the ML practitioner), and our framework can be used to effectively audit data across various domains.

2.2 Membership Inference

Membership inference (MI) is a type of confidentiality attack in machine learning, which aims to infer if a particular data sample [7, 13, 27, 60, 78] or any data associated with a specific user [11, 47, 65] has been used to train a target ML model. The existing MI methods can be classified into shadow model-based attacks [40, 60] and metric-based attacks [56, 58, 66, 79]. Shadow model-based attacks leverage shadow models (i.e., models trained on datasets that are similar to

the training dataset of the target model) to imitate the target model and so incur high costs to train them. In contrast, metric-based attacks leverage metrics that are simple to compute (e.g., entropy of the confidence vector output by the target classifier [58, 66]) while achieving comparable inference performance [56, 58, 79]. MI has been explored for various model types, e.g., image classifiers [56, 58, 66, 79], visual encoders trained by self-supervised learning [39], language models [50], reinforcement learning [19], and facial recognition models [11].

MI can be used as a *passive* data auditing method that a data owner can use to infer if her data is used in an ML model. However, such a passive method does not provide any quantitative guarantee for its inference results. Our proposed framework uses metric-based MI to design the score function in the detection algorithm that provides a quantifiable, tunable guarantee on false detection.

2.3 Data Watermarking

Data watermarking is a technique used to track digital data by embedding a watermark that contains identifying information of the data owner. A classical example of image watermarking is zero-bit watermarking [9] that embeds information into the Fourier transform of the image. However, this type of traditional watermarking is not robust to data transformation. Recently, there have been research efforts on training deep neural networks (DNNs) to embed and recover watermarks that are robust to data transformation [4, 43, 68, 84]. DNN-based data watermarking is widely applied to attribute AI-generated content [21, 81].

Data watermarking can be used to audit data use to train a generative model [81], since the watermark embedded in the training images could be transferred to the images generated from the model. However, this technique cannot be directly applied to other types of ML models, e.g., an image classifier. In contrast, instead of recovering the embedded marks from the ML model, our proposed auditing method detects the use of published data by analyzing the outputs of the ML model on the published data and the hidden data.

3 Problem Formulation

We consider two parties: a *data owner* and a *machine learning practitioner*. The data owner holds a set $\{x_1, x_2, \ldots, x_N\}$ of data that will be published online, e.g., posted on social media to attract attention. The ML practitioner aims to train a machine learning model f of good utility on a set of training data $\mathcal{D} = \{a_i\}_{i=1}^{M}$ of size M by solving:

$$\min_f \frac{1}{M} \sum_{i=1}^{M} \ell(f, a_i), \tag{1}$$

where ℓ is a loss function used to measure the performance of the ML model on the training samples. The definition of the loss function depends on the machine learning task. For example, the loss function in image classification is the cross-entropy loss [45].

3.1 Threat Model

The ML practitioner wants to assemble a training dataset $\mathcal{D}$ that can be used to train a useful ML model. He does so by collecting the data published online from multiple data owners, *without* their authorization. As such, a data owner's data constitutes a subset of the ML practitioner's collected dataset (i.e., some portion of $\{x_1, x_2, \ldots, x_N\}$ or its published version is contained in $\mathcal{D}$). The ML practitioner preprocesses the collected data (e.g., labeling it, if needed), trains an ML model on the preprocessed data using a learning algorithm specified for his ML task (e.g., supervised learning for image classification), and deploys it to provide service to consumers.

The data owner wants to detect the ML practitioner's use of her data. To do so, the data owner needs to apply a method to audit the ML practitioner's ML model such that if the ML model uses her published data, then she will detect this fact from the deployed model. We allow the data owner only black-box access to the deployed ML model. In other words, she does not necessarily know the architecture and parameters of the ML model, but can obtain the outputs of the ML model by providing her queries, e.g., predictions or vectors of confidence scores output by an image classifier given her images as inputs.

3.2 Design Goals

In this work, we aim to design a *data auditing* framework for a data owner, which she can apply to detect the ML practitioner's use of her data. We have the following design goals for the proposed data auditing framework:

- **Effectiveness**: The main goal of the proposed data auditing framework is to detect the unauthorized use of data in ML model training. When the published data is used, the proposed method should successfully detect the use of the owner's data. More specifically, the detection success rate (i.e., the probability of successfully detecting the data use) should grow with the amount of the owner's data that the ML practitioner uses in training, and should approach 100% if most of her data is used.
- **Quantifiable false-detection rate**: When the ML practitioner does *not* use the owner's data, then detection should occur with only a quantifiable probability (e.g., $\leq 5\%$). Such false-detection rate guarantees that if the ML practitioner does not use the data owner's data, then the risk of falsely accusing him is small and quantifiable.
- **Generality**: Once the data owner publishes her data online, the ML practitioner might collect them, label them if needed, and use them in the ML-model training for his designed ML task. The generality goal is that the algorithm applied prior to data publication (i.e., the data-marking algorithm, introduced in Sec. 4.1) should be agnostic to the data labeling and the ML task, and that the proposed data auditing framework can be applied to effectively audit data in any type of ML model (e.g., image classifier or language model).
- **Robustness**: Once the ML practitioner realizes that the data auditing method is applied, he would presumably deploy countermeasures/adaptive attacks to defeat the data auditing method without sacrificing the utility of the trained ML model significantly. The robustness goal requires that the proposed framework is still effective to detect the unauthorized use of data in model training even when utility-preserving countermeasures/adaptive attacks have been applied by the ML practitioner.

4 The Proposed Framework

In this section, we propose a framework used to detect if an ML model has been trained on the data owner's data. In our framework, the data owner does not publish her data $\{x_1, x_2, \ldots, x_N\}$ directly. Instead, she creates two different marked versions of each x_i, namely x_i^0 and x_i^1; uniformly randomly chooses a bit $b_i \xleftarrow{\$} \{0,1\}$; and publishes $x_i^{b_i}$ while keeping $x_i^{1-b_i}$ private. If the ML practitioner's ML model f is *not* trained on the published data, it will behave equally when provided the published data and the unpublished data as input (e.g., for classification). Otherwise, its behavior will be biased towards the published data due to their memorization in training [10, 64].

Formally, let g^f denote a score function g with oracle (i.e., black-box) access to ML model f and that is designed for black-box membership inference [13, 39, 66], so that its output (a real number) indicates the likelihood that its input was a training sample for f. If the ML model f is *not* trained on the published data $x_i^{b_i}$, then the probability of the event $g^f(x_i^{b_i}) > g^f(x_i^{1-b_i})$ will be $\frac{1}{2}$; otherwise, the probability will be larger than $\frac{1}{2}$. The probability $\frac{1}{2}$ is due to the uniformly random sampling of b_i. As such, we can detect if an ML model is trained on a dataset containing a subset of published data by observing the different performance of the ML model on the published data and the unpublished data. Since we compare the membership inference scores (likelihoods) of published data and unpublished data, we refer to this technique as using *contrastive* membership inference. When the published data is used in training, a "useful" membership inference will give a higher score to published data than to unpublished data, even though both scores might be high enough to predict them as "members" independently. More details on how to generate the published data and the unpublished data and how to measure the bias in the ML model will be discussed later (see Sec. 4.1 and Sec. 4.2, respectively).

Generally, our framework includes a marking algorithm and a detection algorithm. The marking algorithm is applied in the marking step before the data publication, while the detection algorithm is applied in the detection step after the ML model deployment.

4.1 Data Marking

The marking algorithm, applied in the marking step, is used to generate a pair of published data and unpublished data. Formally, the marking algorithm takes as input a raw datum x_i, and outputs its published version $x_i^{b_i}$ and its unpublished version $x_i^{1-b_i}$. The marking algorithm includes a *marked data generation* step and a *random sampling* step, and its pseudocode is presented in the full paper [28, App. A]. The marked data generation step creates a pair (x_i^0, x_i^1), both crafted from the raw datum x_i. Taking the example where x_i is an image, we set $x_i^0 \leftarrow x_i + \delta_i$ and $x_i^1 \leftarrow x_i - \delta_i$ where δ_i is the added mark. The random sampling step selects $b_i \xleftarrow{\$} \{0,1\}$ and publishes $x_i^{b_i}$, keeping $x_i^{1-b_i}$ secret.

Basic requirements. We have the following requirements for the generated x_i^0 and x_i^1: *utility preservation* and *distinction*.

- *Utility preservation*: x_i^0 and x_i^1 should provide the same utility as x_i to the data owner, for the purposes for which the data owner wishes to publish x_i (e.g., to attract attention on social media). Formally, given a well-defined distance function $u(\cdot,\cdot)$ measuring the utility difference, utility preservation requires that $u(x_i^0, x_i) \le \epsilon$ and $u(x_i^1, x_i) \le \epsilon$, where ϵ is a small scalar. Taking the example of images, the utility distance function could be defined as the infinity norm of the difference in the pixel values, i.e., $u(x_i^0, x_i) = \left\|x_i^0 - x_i\right\|_\infty$ and $u(x_i^1, x_i) = \left\|x_i^1 - x_i\right\|_\infty$.
- *Distinction*: x_i^0 and x_i^1 should be different enough such that contrastive membership inference can distinguish between a model trained on one but not the other. Formally, given a well-defined distance function $d(\cdot,\cdot)$, distinction requires that $d(x_i^0, x_i^1)$ is maximized. Continuing with the example of images, we could define $d(x_i^0, x_i^1) = \left\|h(x_i^0) - h(x_i^1)\right\|_2$, where h is an image feature extractor, e.g., ResNet18 [25] pretrained on ImageNet [16].[1]

There exists a tension between utility preservation and distinction. Specifically, when the marked data preserves more utility of the raw data, i.e., by using a smaller ϵ, the difference between the two marked versions is smaller and thus it is harder for contrastive membership inference to distinguish between a model trained on one but not the other. In experiments in Sec. 5 and Sec. 6, we show that we can balance utility preservation and distinction well, by setting an appropriate ϵ. We also analyze and discuss this tension in Sec. 5.3.

Marked data generation. To craft a pair of marked data that satisfy the basic requirements, we formulate an optimization problem:

$$\max_{x_i^0, x_i^1} \quad d(x_i^0, x_i^1) \tag{2a}$$

$$\text{subject to:} \quad u(x_i^0, x_i) \le \epsilon \quad \text{and} \quad u(x_i^1, x_i) \le \epsilon \tag{2b}$$

The definitions of $u(\cdot,\cdot)$ and $d(\cdot,\cdot)$, and how to solve Eq. (2) depend on the type of data. We will instantiate them in our experiments in Sec. 5 and Sec. 6.

Random sampling. After crafting the marked data (x_i^0, x_i^1), the data owner selects $b_i \xleftarrow{\$} \{0,1\}$. Then she publishes $x_i^{b_i}$, e.g., on social media. She keeps $x_i^{1-b_i}$ secret to use in the detection step, as discussed in Sec. 4.2.

4.2 Data-Use Detection

The detection algorithm, applied in the detection step, is used to detect if a target ML model is trained on a dataset containing the published data. Formally, given oracle (black-box) access to an ML model f, the detection algorithm takes as input the data owner's published data $\{x_i^{b_i}\}_{i=1}^N$ and her unpublished data $\{x_i^{1-b_i}\}_{i=1}^N$, and outputs a Boolean value. It detects the difference between the outputs of the target ML model on the published data and unpublished data. Specifically, for a given i, the data owner measures if

$$g^f(x_i^{b_i}) > g^f(x_i^{1-b_i}), \tag{3}$$

where the score function g is a black-box membership inference algorithm that measures the likelihood of the input being used as a member of the training set of the target ML model f. A higher score returned by the score function indicates a higher likelihood, and thus the choice of the score function depends on the type of

[1] An image feature extractor is not necessary but helpful to craft marked images. Our proposed method can audit image data effectively even if no image feature extractor is used in marked data generation, as shown in the full paper [28, App. G].

the target ML model; we will give examples in Sec. 5 and Sec. 6. Under the null hypothesis H_0 that the ML model f was not trained on the data owner's published data, Eq. (3) holds with probability $\pi = \frac{1}{2}$, where the probability is with respect to the choice of b_i. If it was trained on the data owner's published data (the alternative hypothesis H_1), however, then it is reasonable to expect that Eq. (3) holds with probability $\pi > \frac{1}{2}$, since the ML model memorizes the published data. As such, the detection problem can be formulated to test the following hypothesis:

- Null hypothesis H_0: $\pi = \frac{1}{2}$.
- Alternate hypothesis H_1: $\pi > \frac{1}{2}$.

We denote the sum of successful measurements in the population as N', i.e., $N' = \sum_{i=1}^{N} \mathbb{I}(g^f(x_i^{b_i}) > g^f(x_i^{1-b_i}))$ where $\mathbb{I}$ is the indicator function returning 1 if the input statement is true or returning 0 if the input statement is false. Under H_0, N' follows a binomial distribution with parameters N and $p' = \frac{1}{2}$. As such, the data owner can reject H_0 or not based on the measured N' using a binomial test. In other words, the data owner detects if the ML model is trained on her published data according to N'.

4.2.1 Estimate N' by Sampling Sequentially WoR. Measuring N' exactly requires querying all the published data and hidden data to the ML model, e.g., via its API interface. When N is large, this would be highly costly and time consuming. To address this, we apply a sequential method: at each time step, the data owner samples an i uniformly at random without replacement (WoR) and estimates N' based on the currently obtained measurements. The classical sequential hypothesis testing method, namely the sequential probability ratio test [73], requires knowing the probability π in the alternate hypothesis H_1 and so does not fit our problem.

Sampling WoR problem. There are N fixed but unknown objects in the finite population $\{I_1, \ldots, I_N\}$, where each I_i takes on a value in $\{0, 1\}$, specifically $I_i = \mathbb{I}(g^f(x_i^{b_i}) > g^f(x_i^{1-b_i}))$. The data owner observes one object per time step by sampling it uniformly at random WoR from the population, so that:

$$\mathbb{I}_t \mid \{\mathbb{I}_1, \ldots, \mathbb{I}_{t-1}\} \sim \mathsf{Uniform}(\{I_1, \ldots, I_N\} \setminus \{\mathbb{I}_1, \ldots, \mathbb{I}_{t-1}\}),$$

where $\mathbb{I}_t$ denotes the object sampled at time $t \in \{1, 2, \ldots, N\}$. As such, the variable $\mathbb{N}_t = \sum_{n=1}^{t} \mathbb{I}_n$ at time t ($t \leq N$) follows a hypergeometric distribution:

$$\mathbb{P}\left(\mathbb{N}_t = N''\right) = \binom{N'}{N''}\binom{N - N'}{t - N''} \Big/ \binom{N}{t},$$

where $N'' \in \{0, 1, \ldots, \min(N', t)\}$ is the number of ones from the obtained observations at t, and $\binom{N'}{N''}$ denotes N' choose N''.

Estimate N' by prior-posterior-ratio martingale (PPRM) [75]. In the above problem of sampling WoR from a finite population, the data owner can use a prior-posterior-ratio martingale (PPRM) [75] to obtain a confidence interval $C_t(\alpha) = [L_t(\alpha), U_t(\alpha)]$ for N' at the time t, which is a function of the confidence level α, e.g., $\alpha = 0.05$. Such a sequence of confidence intervals $\{C_t(\alpha)\}_{t \in \{1,2,\ldots,N\}}$ has the following guarantee [75]:

$$\mathbb{P}\left(\exists t \in \{1, 2, \ldots, N\} : N' \notin C_t(\alpha)\right) \leq \alpha.$$

In words, the probability that there exists a confidence interval where N' is excluded is no larger than α.

4.2.2 Detection Algorithm with Quantifiable False-Detection Rate. We present the pseudocode of our detection algorithm in the full paper [28, App. A]. At each time step, the data owner samples an $i \in \{1, \ldots, N\}$ uniformly at random WoR and estimates N' based on the currently obtained measurements using a prior-posterior-ratio martingale (PPRM) [75] that takes as inputs the sequence of measurements so far, the size of the population N, and the confidence level α. It returns a confidence interval for N'. If the interval (i.e., its lower bound) is equal to or larger than a preselected threshold T, the data owner stops sampling and rejects the null hypothesis; otherwise, she continues the sampling.

Since the detection algorithm rejects the null hypothesis as long as the lower bound of a confidence interval is equal to or larger than a preselected threshold T, the false-detection probability is $\mathbb{P}\left(\exists t \in \{1, 2, \ldots, N\} : L_t(\alpha) \geq T \;\middle|\; H_0\right)$. We prove the following theorem in the full paper [28, App. B].

Theorem 1 (False detection rate). *For $T \in \{\lceil \frac{N}{2} \rceil, \ldots, N\}$ and $\alpha < p$ such that $\left(\frac{\exp\left(\frac{2T}{N} - 1\right)}{\left(\frac{2T}{N}\right)^{\frac{2T}{N}}}\right)^{\frac{N}{2}} \leq p - \alpha$, our data-use detection algorithm has a false-detection rate less than p. In other words:*

$$\mathbb{P}\left(\exists t \in \{1, 2, \ldots, N\} : L_t(\alpha) \geq T \;\middle|\; H_0\right) < p.$$

5 Auditing Image Classifiers

In this section, we apply our data-use auditing method to detect unauthorized use of data to train an image classifier. Image classification (e.g., [16, 25, 61]) is a fundamental computer-vision task in which the ML practitioner trains a model (i.e., image classifier) on training data partitioned into J classes. For a newly given image, the ML model predicts a class label for it or, more generally, a vector of J dimensions. The output vector could be a vector of confidence scores whose j-th component represents the probability of the input being from the j-th class, or a one-hot vector where only the component of the predicted class is 1 and the others are 0. Each training sample in the training set $\mathcal{D}$ is an (image, label) pair, where the image might be collected online and the label is assigned by the ML practitioner after the data collection. The loss function in Eq. (1) is the cross-entropy loss [45].

5.1 Score Function

Here we define the score function g^f used in our detection algorithm for the image classifier f. The score function is a black-box membership inference test based on the intuition that the ML model is more likely to output a confident and correct prediction for a perturbed training sample than for a perturbed non-training sample. This basic idea is similar to existing label-only membership inference methods (e.g., [13]). The confidence and correctness of the output are measured by entropy [58] or modified entropy [66] if the ground-truth label of the input is known. Specifically, we define the score function as follows: given an input image, we first randomly generate K perturbed versions, and then obtain K outputs using the perturbed images as inputs to the target ML model. We average the K outputs and use the negative (modified) entropy of the averaged output vector elements as the score. The details of the score function are shown in the full paper [28, App. C].

5.2 Experimental Setup

Datasets. We used three image benchmarks: CIFAR-10 [32], CIFAR-100 [32], and TinyImageNet [33]:

- **CIFAR-10**: CIFAR-10 is a dataset containing 60,000 images of 3×32×32 dimensions partitioned into $J = 10$ classes. In CIFAR-10, there are 50,000 training samples and 10,000 test samples.
- **CIFAR-100**: CIFAR-100 is a dataset containing 60,000 images of $3 \times 32 \times 32$ dimensions partitioned into $J = 100$ classes. In CIFAR-100, there are 50,000 training samples and 10,000 test samples.
- **TinyImageNet**: TinyImageNet is a dataset containing images of $3 \times 64 \times 64$ dimensions partitioned into $J = 200$ classes. In TinyImageNet, there are 100,000 training samples and 10,000 validation samples that we used for testing.

Marking setting. In each experiment, we uniformly at random sampled N samples $\{x_i\}_{i=1}^N$ from the training sample set $\mathcal{X}$ of a dataset. The N samples are assumed to be owned by a data owner. Here we set $\frac{N}{|\mathcal{X}|} = 10\%$ as the default, i.e., $N = 5{,}000$ for CIFAR-10 or CIFAR-100, and $N = 10{,}000$ for TinyImageNet. We applied our data marking algorithm to generate the published data $\{x_i^{b_i}\}_{i=1}^N$ and the unpublished data $\{x_i^{1-b_i}\}_{i=1}^N$ for $\{x_i\}_{i=1}^N$. In Eq. (2), we used $\epsilon = 10$ as the default when the pixel range of image is $[0, 255]$. We defined the two marked versions by $x_i^0 \leftarrow x_i + \delta_i$ and $x_i^1 \leftarrow x_i - \delta_i$ (δ_i is the mark), utility distance function by $u(x_i^0, x_i) = \left\|x_i^0 - x_i\right\|_\infty$ and $u(x_i^1, x_i) = \left\|x_i^1 - x_i\right\|_\infty$, and the distance function by $d(x_i^0, x_i^1) = \left\|h(x_i^0) - h(x_i^1)\right\|_2$, where we used ResNet18 [25] pretrained on ImageNet [16] to be the default feature extractor h. We solved Eq. (2) by projected gradient descent [37]. Then we uniformly at random sampled a subset (of size $\hat{N}$) of $\{x_i^{b_i}\}_{i=1}^N$ as $\hat{\mathcal{X}}$ (i.e., $\hat{\mathcal{X}} \subseteq \{x_i^{b_i}\}_{i=1}^N$) to simulate a general case where the ML practitioner collected a subset of published data as training samples. By default, we set $\hat{N} = N$. As such, we constituted the training dataset collected by the ML practitioner as $\mathcal{D} = (\mathcal{X} \setminus \{x_i\}_{i=1}^N) \cup \hat{\mathcal{X}}$ with correct labels (i.e., using the same labels as those in the dataset). Some examples of marked images are displayed in the full paper [28, App. D].

Training setting. We used ResNet18 as the default architecture of the ML model f trained by the ML practitioner. We used a standard SGD algorithm to train f, as follows: f was trained on normalized training data with default data augmentation applied [22] using an SGD optimizer [3] with a weight decay of 5×10^{-4} for 80 epochs, a batch size of 128, and an initial learning rate of 0.1 decayed by a factor of 0.1 when the number of epochs reached 30, 50, or 70.

Detection setting. In each detection experiment, we applied our data-use detection algorithm to the given ML model f using a set of pairs of generated published data and unpublished data. In the data-use detection algorithm and the score function, we set $\alpha = 0.025$, $p = 0.05$, and $K = 16$ as the default. (Recall from Thm. 1 that p bounds the false-detection rate.) We present results for four different experimental conditions that define the information available to the detector, denoted as CG, C$\bar{\mathrm{G}}$, $\bar{\mathrm{C}}\mathrm{G}$, and $\bar{\mathrm{C}}\bar{\mathrm{G}}$. We define these four conditions in Table 1.

Condition	Confidence score	Ground-truth
CG	✔	✔
C$\bar{\mathrm{G}}$	✔	✗
$\bar{\mathrm{C}}\mathrm{G}$	✗	✔
$\bar{\mathrm{C}}\bar{\mathrm{G}}$	✗	✗

Table 1: Information available to the detector. "Confidence score" indicates whether the ML model f outputs a full confidence vector ("✔") or just a label, i.e., a one-hot vector ("✗"). "Ground-truth" indicates whether the true label of a query to the ML model is known by the detector ("✔") or not ("✗").

Baselines. We used two state-of-the-art methods, Radioactive Data [55], which we abbreviate to RData, and Untargeted Backdoor Watermark-Clean (UBW-C) [35], as baselines. RData requires knowledge of the class labels for its data. So, we also consider two variants of RData in which the data owner is presumed to not know how the ML practitioner will label her data, and so applies the same mark to all of her data regardless of class ("RData (one mark)"), or to know only a "coarse" label (superclass) of the class label the ML practitioner will assign to each ("RData (superclass)"). The details of baselines and their implementation are described in the full paper [28, App. E].

Metrics. We used the following metrics to evaluate the methods:

- **Test accuracy** (acc): acc is the fraction of test samples that are correctly classified by the ML model f. A higher acc indicates a better performance of the ML model.
- **Detection success rate** (DSR): DSR is the fraction of detection experiments returning True (i.e., affirmatively detecting data use). When the detected ML model did use the published data, a higher DSR indicates a better performance of the data auditing framework. When the detected ML model did not use the published data, a lower DSR indicates more robustness to false detections.
- **Minimum amount of published data used in training, as a percentage of the training data set, to trigger detection** (P): That is, P is the minimum value of $\hat{N}/M$, expressed as a percentage, at which the detection algorithm returns True. Therefore, a lower P indicates a more sensitive detector. However, to find P in each of our settings is costly since we need to exhaustively test potential values of $\hat{N}/M$. For this reason, we report an alternative measure (see below) in place of P.
- **Query cost** (cost): cost is the number of queries to the target ML model f to conclude that f was trained on the data-owner's data. That is, cost $= 2 \times K \times Q$, where Q ($Q \leq N$) is the number of published data used to query the ML model to detect its training with the data-owner's data. It indicates the practical cost used in the detection step. A lower cost indicates a more cost-efficient detection method.
- **Ratio between the number of queried published data and the total number of training samples** ($\frac{Q}{M}$): $\frac{Q}{M}$ is the ratio between the number Q of published data used to query the ML model (resulting in detection) and the total number M of training samples. In our tests, $\frac{Q}{M}$ was strongly correlated with P (a Pearson correlation coefficient [15] of 0.66 with high statistical significance; see the full paper [28, App. B]) and is considerably

cheaper to compute than P. Moreover, for a fixed K and $\mathcal{D}$, cost is a linear function of $\frac{Q}{M}$. Therefore, when presenting our results, we use $\frac{Q}{M}$ as a surrogate for P and cost. A lower $\frac{Q}{M}$ indicates a lower P and a lower cost, and thus it suggests a more detection-efficient and more cost-efficient method.

5.3 Experimental Results

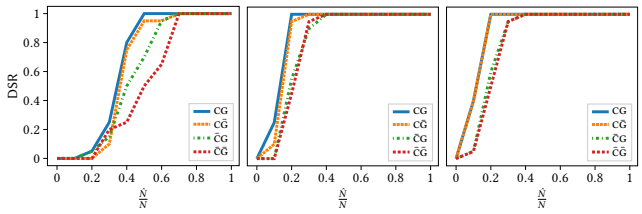
5.3.1 *Overall Performance.*

Effectiveness. The detection performance of our proposed method on different visual benchmarks is shown in Table 2. Table 2 demonstrates that our method is highly effective to detect the use of published data in training ML models, i.e, yielding a 20/20 DSR in all settings where the published data is used as a subset of training samples of the target ML model. In addition, the ML models trained on the datasets including the published data preserved good utility, i.e., their acc values are only slightly lower (< 1% on average) than those trained on clean datasets. For detection, we needed a $\frac{Q}{M}$ ranging from 2.20% to 4.65% for CIFAR-10, from 0.19% to 0.60% for CIFAR-100, and from 0.14% to 0.67% for TinyImageNet. These results show that our method achieved more detection efficiency when applied to a classification task with a large number of classes. Such ranges of $\frac{Q}{M}$ also indicate that detection needs a number of queries to the ML model (i.e., cost) ranging from a hundred to tens of thousands. Given the current prices of online queries to pretrained visual AI models (e.g., \$1.50 per 1,000 images[2]), the detection cost is affordable, ranging from several dollars to a hundred dollars. When we have less information on the output of the ML model (i.e., the outputs are the predictions only) or the queries (i.e., the ground-truth labels are unknown) in the detection, we needed more queries to trigger detection, i.e., yielding a larger $\frac{Q}{M}$.

Impact of using published data partially and false-detections. After the published data is released online, the ML practitioner might collect them partially (i.e., $\frac{\hat{N}}{N}$ is smaller than 1.0) and use the collected data in training. Here, we tested the detection performance of our method on the ML model trained on $\mathcal{D}$ under different ratios of $\frac{\hat{N}}{N}$. The results are shown in Fig. 1. When the ML practitioner used more published data, DSR was higher. Especially, when he used $\geq$ 70% published CIFAR-10 data, or $\geq$ 40% published CIFAR-100 data or published TinyImageNet data, we achieved a DSR of 20/20, even with the least information (condition $\bar{c}\bar{G}$). When the ML practitioner did not use any published data in training (i.e., $\frac{\hat{N}}{N} = 0$), DSR was 0/20 under all considered settings, which empirically confirms the upper bound $p = 0.05$ on false-detection rate of our method.

Comparison with baselines. Table 3 summarizes the comparison between our method and baselines. Compared with the baselines, our method is more effective in the detection of data use, i.e., yielding a higher DSR and a higher acc. More importantly, different from the two state-of-the-art methods (i.e., RData and UBW-C), our method does not need the labeling of training samples before data publication or the white-box access to the ML model (i.e., knowing the parameters of the ML model). The variants of RData denoted as

[2]https://cloud.google.com/vision/pricing



(a) CIFAR-10 (b) CIFAR-100 (c) TinyImageNet

Figure 1: The impact of $\frac{\hat{N}}{N}$ on the detection performance (the default $\frac{\hat{N}}{N}$ is 1.0). The results from $\frac{\hat{N}}{N} = 0$ are the false-detections of our method.

"one mark" and "superclass" do not need the complete information on labeling, but their DSR dropped significantly.

Multiple data owners. Here we consider a general real-world setting where there are multiple data owners applying data auditing independently, each of which set the upper bound on the false-detection rate as $p = 0.05$. In these experiments, each data owner had 5,000 CIFAR-100 data items (i.e., 10% of the training samples collected by the ML practitioner) to publish. Each applied an auditing framework to generate her marked data and to detect its use in the deployed ML model independently. The detection results with our method and with the state-of-the-art method, RData (with full information on data labeling), are shown in Table 4. Compared with RData, whose detection performance degraded with a larger number of data owners, our method was much more effective, yielding a 100% DSR in all cases.

The results in Table 2, Fig. 1, Table 3, Table 4 demonstrate that our method achieves our *effectiveness* goal defined in Sec. 3.2. Table 3 and Table 4 show the advantages of our proposed method over the baselines. Table 4 presents interesting results under real-world settings where multiple data owners independently audit an ML model for use of their data.

5.3.2 *Impact of ML Model Architecture and Hyperparameters.*

In this section, we explore the impact of the ML practitioner's model architecture and the data owner's hyperparameters on detection, such as the utility bound ϵ, the feature extractor h used to generate marked data, the upper bound p on the false-detection rate, and the number K of sampled perturbations per image in detection. Due to the space limit, we present results in the full paper [28, App. G].

5.3.3 *Robustness to Countermeasures/Adaptive Attacks.*

When the ML practitioner knows that a data owner marked her data, he might utilize countermeasures/adaptive attacks to defeat the auditing method. His goal is to decrease DSR without degrading the performance of the trained ML model significantly. We evaluated the robustness of the proposed method to three types of countermeasures/adaptive attacks, described below.

Limiting the information from the ML model output. Since our detection method measures the difference between outputs of the ML model on the published data and unpublished data, the ML practitioner can limit the output (e.g., the vector of confidence scores) of the deployed ML model, aiming to degrade our detection. Here we considered two countermeasures of this type:

	acc%	△acc%	CG		CḠ		C̄G		C̄Ḡ	
			DSR	$\frac{Q}{M}$	DSR	$\frac{Q}{M}$	DSR	$\frac{Q}{M}$	DSR	$\frac{Q}{M}$
CIFAR-10	93.64	−0.05	20/20	2.20%	20/20	2.67%	20/20	4.22%	20/20	4.65%
CIFAR-100	74.29	−0.76	20/20	0.19%	20/20	0.20%	20/20	0.59%	20/20	0.60%
TinyImageNet	59.13	−0.16	20/20	0.14%	20/20	0.13%	20/20	0.59%	20/20	0.67%

Table 2: Overall performance of our proposed method on different image benchmarks, with an upper bound of $p = 0.05$ on the false-detection rate. All results are averaged over 20 experiments. The numbers in the △acc% column are the differences between averaged accuracies of ML models trained on marked datasets and those of ML models trained on clean datasets.

		Labeling known	White box	Bounded FDR	1%		2%		5%		10%	
					DSR	acc %	DSR	acc %	DSR	acc %	DSR	acc %
CIFAR-10	Our method (C̄Ḡ)	○	○	✔	**8/20**	**93.79**	**11/20**	**93.71**	**19/20**	**93.70**	**20/20**	**93.64**
	RData	●	●	✔	1/20	93.65	2/20	93.56	2/20	93.29	4/20	93.26
	RData (one mark)	○	●	✔	0/20	93.75	0/20	93.60	0/20	93.42	0/20	93.25
	UBW-C ($\tau = 0.25$)	●	○	✗	0/20	93.50	0/20	93.14	0/20	92.67	2/20	92.73
	UBW-C ($\tau = 0.20$)	●	○	✗	1/20	93.50	8/20	93.15	7/20	92.46	15/20	92.52
CIFAR-100	Our method (C̄Ḡ)	○	○	✔	**20/20**	**75.01**	**20/20**	**74.94**	**20/20**	**74.60**	**20/20**	**74.29**
	RData	●	●	✔	5/20	74.66	14/20	74.57	**20/20**	73.81	**20/20**	73.53
	RData (superclass)	◐	●	✔	4/20	74.76	10/20	74.46	14/20	73.99	19/20	73.42
	RData (one mark)	○	●	✔	0/20	74.70	0/20	74.51	1/20	74.05	0/20	73.51
	UBW-C ($\tau = 0.25$)	●	○	✗	0/20	74.60	0/20	74.16	16/20	73.30	**20/20**	72.32
	UBW-C ($\tau = 0.20$)	●	○	✗	19/20	74.60	**20/20**	74.33	**20/20**	73.21	**20/20**	72.47
TinyImageNet	Our method (C̄Ḡ)	○	○	✔	**20/20**	**59.32**	**20/20**	**59.24**	**20/20**	**59.17**	**20/20**	**59.13**
	RData	●	●	✔	8/20	59.14	18/20	58.94	**20/20**	58.59	**20/20**	58.13
	RData (superclass)	◐	●	✔	7/20	59.14	14/20	59.03	**20/20**	58.71	**20/20**	58.09
	RData (one mark)	○	●	✔	2/20	59.12	1/20	58.98	0/20	58.61	0/20	58.29
	UBW-C ($\tau = 0.25$)	●	○	✗	0/20	59.01	0/20	58.80	0/20	58.43	0/20	57.78
	UBW-C ($\tau = 0.20$)	●	○	✗	0/20	59.01	0/20	58.62	6/20	58.41	17/20	57.63

Table 3: Comparison between our proposed method and baselines under different rates of $\frac{\hat{N}}{M} \in \{1\%, 2\%, 5\%, 10\%\}$. The results of our method come from the setting with least information available to the data owner, i.e., C̄Ḡ. In UBW-C, τ is a hyperparameter of its detection algorithm. In the columns of "Labeling known" and "White box", "●" indicates that the information is needed; "○" means that information is not needed; "◐" means that partial information is needed. In the column "bounded FDR", "✔" ("✗") indicates that the method provides (does not provide) a provable bound on the false-detection rate. Results are averaged over 20 experiments. The bold results are the best ones among the compared methods.

	Data owners			
	1	2	5	10
Our method (C̄Ḡ)	20/20	40/40	100/100	200/200
RData	20/20	38/40	64/100	90/200

Table 4: Comparison between our method and RData (which requires knowledge of data labeling), both under an upper bound of $p = 0.05$ on the false-detection rate, when multiple data owners applied data auditing independently. Each owner contributed 10% of the training dataset. Results are the total detections over all detection attempts (by all data owners) in 20 experiments.

- Outputting only the top κ confidence scores (Topκ): This countermeasure allows the deployed ML model to output the top κ confidence scores, masking out the others in the output vector. Here we considered $\kappa = 1$ and $\kappa = 5$.
- Adding perturbation into the output (MemGuard [29]): This countermeasure adds carefully crafted perturbations into the ML model output to limit the information given. We considered MemGuard proposed by Jia, et al. [29] to design the perturbation, where we used a moderate distortion budget of 0.5.

Note that these countermeasures can be applied only in the ML model deployment where the output is a vector of confidence scores instead of a prediction/label.

Reducing memorization of training samples. Intuitively, the ML practitioner can apply methods to discourage memorization of training samples by the ML model, so that the published data and the unpublished data will have similar scores by the defined score function. As such, reducing memorization of training samples could render the detection method to be less effective. We considered three such countermeasures:

- Differential privacy (DP [20]): DP is a standard privacy definition that limits the information leaked about any training input in the

output of the algorithm. To achieve DP, the ML practitioner clips the gradients of each training batch and adds Gaussian noise (with standard deviation of σ) into the clipped gradients during ML model training [2].
- Early stopping (EarlyStop): In this countermeasure, the ML practitioner trains the ML model for a small number of epochs to prevent the ML model from overfitting to the training samples. Here we trained ML models for 20, 40, and 60 epochs, denoted as EarlyStop(20), EarlyStop(40), and EarlyStop(60), respectively.
- Adversarial regularization (AdvReg [49]): Adversarial regularization is a strategy to generalize the ML model. It does so by alternating between training the ML model to minimize the classification loss and training it to maximize the gain of a membership inference attack. In the implementation of AdvReg, we set the adversarial regularization factor to be 1.0 [49].

Other attacks. We also considered some other adaptive attacks that aim to defeat our auditing method:
- Detecting pairs of published data and unpublished data in queries (PairDetect): The intuition behind this pair detection is that if the deployment can detect queries of a pair of published data and unpublished data, then it will return the same output to evade detection. We design such a pair detection method as follows: we maintain a window of queries in the history and their ML model outputs, and we compare each new query with those in the window to decide what to output. If the infinity norm of the pixel difference between the new query and a previous query is smaller than 2ϵ, we return the output of the previous query; otherwise, we return the output for the new query.
- Adding Gaussian noise into the training samples (Gaussian(σ)): This method adds noise into each training sample to mask the added mark. The added noise is sampled from a Gaussian distribution with standard deviation σ.
- Avoiding data augmentation in training (NoTrainAug): Excluding data augmentation in ML model training will degrade the effectiveness of the label-only membership inference that we apply as the score function for the image classifier, as demonstrated by previous works (e.g., [13]).
- Using our marking algorithm (with the default hyperparameters) to perturb training samples (MarkPerturb): This countermeasure applies our marking algorithm (with the default hyperparameters) to craft two perturbed versions of each training sample and randomly selects one to use in training.[3]

Results. We summarize the robustness of our method to these countermeasures/adaptive attacks in Table 5. As shown in Table 5, masking (Top5, Top1, MemGuard) had limited impact on our detection effectiveness, yielding a slightly higher $\frac{Q}{M}$ but not changing DSR at all. DP and EarlyStop did decrease DSR. However, these countermeasures damaged the utility of the trained ML model, yielding a much smaller acc. Specifically, the application of differential privacy needed a high level of privacy guarantee to defeat our method and so added a large amount of Gaussian noise into the training process to do so. The added noise affected the performance of the ML model, decreasing acc to 64.11%, more than 10 percentage points lower than acc with the default training method. Likewise, to degrade the detection performance of our method, early stopping needed to stop the training when reaching a small number of training epochs, at the cost of low accuracy as well, e.g., acc = 67.10% at 20 epochs. Among the other attacks, detecting queried pairs and excluding data augmentation in ML model training were not useful to counter our method. Pair detection (PairDetect) did not work well to detect queried pairs because we only queried the ML model with their randomly cropped versions, which evaded pair detection. Excluding data augmentation in training did not reduce DSR but diminished the accuracy of the ML model significantly, yielding a low acc of 61.59%. Adding sufficient Gaussian noise to mask the marks before training reduced the detection effectiveness of our method but, again, it also destroyed the utility of the ML model. For example, adding Gaussian noise with $\sigma = 30$ into marked CIFAR-100 data reduced DSR from 20/20 to 6/20 in condition $\bar{c}\bar{G}$ but also decreased acc to 62.10%. The last adaptive attack, i.e., applying our marking algorithm to add perturbations, did not decrease DSR but increased $\frac{Q}{M}$, at the cost of achieving a lower acc of 70.49%. This is because the perturbed published data created by the marking algorithm was still closer to the published data than to the unpublished data, which caused the published data to appear more likely to have been used in the training of the ML model trained on the perturbed published data.

In summary, countermeasures/adaptive attacks we considered in this work did not defeat our auditing method or did so at the cost of sacrificing the utility of the trained ML model; i.e., none achieved a low DSR and a high acc at the same time. Therefore, we conclude that our method achieves the *robustness* goal defined in Sec. 3.2 for image classifiers.

6 Auditing Foundation Models

In this section, we apply our data auditing method to detect unauthorized use of data in foundation models. Foundation models are a class of large, deep neural networks for general-purpose use that are pretrained on large-scale unlabeled data by unsupervised learning or self-supervised learning [6, 12, 18, 52–54, 70]. Examples of foundation models include visual encoders trained by self-supervised learning (e.g., SimCLR [12]), large language models (LLMs) (e.g., ChatGPT [53]), and multimodal models (e.g., CLIP [52]). These models can be used as backbones in various ML tasks, e.g., image classification [16, 25], object detection [83], sentiment analysis [51], text generation [71], and question answering [18], by being fine-tuned on small datasets for these tasks.

We studied the effectiveness of our proposed method on auditing data-use in foundation models by considering three case studies: a visual encoder trained by SimCLR [12], Llama 2 [70], and CLIP [52].

6.1 Visual Encoder

We consider visual encoder, which is a type of foundation model used to learn the general representations of images. A visual encoder can be used as a feature extractor to extract features of images in many vision recognition tasks, e.g., image classification and object detection. A visual encoder is an ML model that takes as input an image and outputs its representation as a feature vector. It is

[3] We could use both perturbed versions in training but we would need to reduce the number of epochs to half (i.e., 40 epochs) for fair comparison.

		acc%	CG		CḠ		ḠG		ḠḠ	
			DSR	$\frac{Q}{M}$	DSR	$\frac{Q}{M}$	DSR	$\frac{Q}{M}$	DSR	$\frac{Q}{M}$
No adaptive attack		74.29	20/20	0.19%	20/20	0.20%	20/20	0.59%	20/20	0.60%
Masking output	Top5	74.29	20/20	0.21%	20/20	0.21%	-	-	-	-
	Top1	74.29	20/20	0.24%	20/20	0.24%	-	-	-	-
	MemGuard	74.29	20/20	1.57%	20/20	1.65%	-	-	-	-
Memorization reduction	DP($\sigma = 0.001$)	70.01	20/20	0.65%	20/20	5.17%	20/20	0.83%	20/20	3.67%
	DP($\sigma = 0.002$)	64.11	20/20	3.98%	1/20	9.99%	20/20	5.74%	2/20	9.97%
	DP($\sigma = 0.003$)	59.25	18/20	8.14%	0/20	10.00%	10/20	9.34%	0/20	10.00%
	EarlyStop(60)	73.50	20/20	0.25%	20/20	0.28%	20/20	0.51%	20/20	0.63%
	EarlyStop(40)	69.15	20/20	0.70%	20/20	3.27%	20/20	1.48%	20/20	3.29%
	EarlyStop(20)	67.10	20/20	3.18%	1/20	10.00%	20/20	5.68%	3/20	9.51%
	AdvReg	60.18	20/20	0.74%	20/20	1.78%	20/20	0.91%	20/20	2.90%
Other attacks	PairDetect	74.29	20/20	0.20%	20/20	0.20%	20/20	0.64%	20/20	0.74%
	NoTrainAug	61.59	20/20	0.19%	20/20	0.30%	20/20	0.51%	20/20	0.85%
	Gaussian($\sigma = 10$)	70.64	20/20	0.40%	20/20	0.45%	20/20	1.72%	20/20	2.08%
	Gaussian($\sigma = 20$)	65.97	20/20	1.02%	20/20	1.56%	20/20	5.31%	19/20	7.12%
	Gaussian($\sigma = 30$)	62.10	20/20	4.03%	20/20	6.64%	16/20	8.93%	6/20	9.90%
	MarkPerturb	70.49	20/20	0.52%	20/20	0.56%	20/20	3.00%	20/20	3.29%

Table 5: Robustness of our proposed method on CIFAR-100 against countermeasures/adaptive attacks. The most effective countermeasure to degrade the detection performance of our method is differential privacy, but it also destroyed the utility of the ML model. All results were averaged over 20 experiments.

trained by self-supervised learning (e.g., SimCLR [12]) on unlabeled data (i.e., each instance in $\mathcal{D}$ is an image). The loss function used by SimCLR is Normalized Temperature-scaled Cross Entropy (NT-Xent) [12].

6.1.1 Score Function. We defined the score function g^f used in the detection algorithm targeting the self-supervised visual encoder using a black-box membership inference method introduced in EncoderMI [39]. The intuition behind it is that the visual encoder f generates more similar feature vectors of two perturbed versions of a training sample than of a non-training sample [39]. In other words, if $x_i^{b_i}$ was used in training f while $x_i^{1-b_i}$ was not, then $\text{cosim}(f(x_{i,1}^{b_i}), f(x_{i,2}^{b_i})) > \text{cosim}(f(x_{i,1}^{1-b_i}), f(x_{i,2}^{1-b_i}))$ where cosim denotes the cosine similarity, $x_{i,1}^{b_i}$ and $x_{i,2}^{b_i}$ are two perturbed versions of $x_i^{b_i}$, and $x_{i,1}^{1-b_i}$ and $x_{i,2}^{1-b_i}$ are two perturbed versions of $x_i^{1-b_i}$. As such, we defined the score function g^f as follows: given an input image, we first randomly generate K of its perturbed versions (e.g., by random cropping and flipping), and then obtain K feature vectors using the perturbed images as inputs to the target visual encoder; second, we compute the cosine similarity of every pairs of feature vectors and return the sum of cosine similarities as the score. The score function g^f is summarized in the full paper [28, App. C].

6.1.2 Experimental Setup.

Datasets. We used three image benchmark datasets: CIFAR-10, CIFAR-100, and TinyImageNet, as introduced in Sec. 5.2.

Marking setting. We followed the setup introduced in Sec. 5.2 to generate the marked dataset, without labels needed. Our using the same marking setup indicates that the application of our marking algorithm is agnostic to the ML task.

Training setting. We followed the previous work (e.g., [12]) to train the ML model by SimCLR, which takes as inputs a base encoder and a projection head (i.e., a multilayer perceptron with one hidden layer). We used ResNet18 as the default architecture of the base encoder. The SimCLR algorithm works as follows: at each training step, we randomly sampled a min-batch (i.e., of size 512) of images from the training set and generated two augmented images from each sampled instance by random cropping and resizing, random color distortion, and random Gaussian blur. The parameters of the base encoder and the projection head were updated by minimizing the NT-Xent loss among the generated augmented images, i.e., maximizing the cosine similarity between any positive pair (i.e., two augmented images generated from the same sampled instance) and minimizing the cosine similarity between any negative pair (i.e., two augmented images generated from different sampled instances). We used SGD with Nesterov Momentum [67] of 0.9 and a weight decay of 10^{-6} as the optimizer, and applied a cosine annealing schedule [41] to update the learning rate, which was set to 0.6 initially. We trained the base encoder and the projection head by 1,000 epochs as the default, and returned the base encoder as the visual encoder f deployed by the ML practitioner.

Detection setting. In the detection algorithm and the score function, we set $\alpha = 0.025$, $p = 0.05$, and $K = 64$ as the default.

Metrics. We used the following metrics for evaluation:

- **Test accuracy of downstream classifier** (acc): acc is the fraction of test samples that are correctly classified by a downstream classifier that uses the visual encoder as the backbone and is

fine-tuned on a small set of data. We followed previous work (e.g., [12]) to fine-tune the downstream classifier on 10% of the clean training samples with their labels (i.e., 5,000 clean CIFAR-10 data, 5,000 clean CIFAR-100 data or 10,000 clean TinyImageNet data). A higher acc indicates a better performance of the visual encoder.

- **Detection success rate** (DSR): please see the description of this metric in Sec. 5.2.
- **Ratio between the number of queried published data and the total number of training samples** ($\frac{Q}{M}$): please see the description of this metric in Sec. 5.2.

6.1.3 Experimental Results. The overall experimental results on three visual benchmarks are presented in Table 6. As shown in Table 6, our proposed method achieved highly effective detection performance on auditing data in visual encoders, yielding a 19/20 DSR for CIFAR-10 and a 20/20 DSR for CIFAR-100 and TinyImageNet.

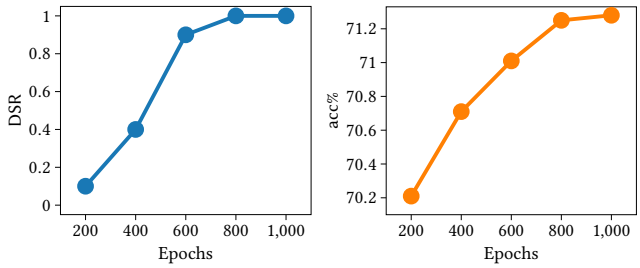
	DSR	$\frac{Q}{M}$
CIFAR-10	19/20	7.12%
CIFAR-100	20/20	7.28%
TinyImageNet	20/20	7.82%

Table 6: Results on auditing data in visual encoder trained by SimCLR, under an upper bound of $p = 0.05$ on the false-detection rate. 10% of training samples were marked. All results were averaged over 20 experiments.

We investigated the impact of training epochs of visual encoder on the detection performance of the auditing method. We trained visual encoders on marked CIFAR-100 by epochs of 200, 400, 600, 800, and 1,000 (1,000 is the default number of epochs). As shown in Fig. 2, when we trained the encoder with a smaller number of epochs, the encoder memorized the training samples less and thus we had a lower DSR. However, training encoder with fewer epochs yielded modestly lower encoder utility, measured by the test accuracy of the downstream classifier (i.e., acc). This suggests that early stopping (i.e., training with a small number of epochs) can degrade the detection performance of our method, but cannot completely alleviate the trade-off between evading detection and encoder utility.

6.2 Llama 2

In this section, we study the application of data auditing to a large language model (LLM). An LLM is a type of large ML model that can understand and generate human language. Here we consider Llama 2 [70] published by Meta AI in 2023, which is an open-sourced LLM with notable performance and, more importantly, is free for research [70]. Specifically, Llama 2 is a family of autoregressive models that generate text by predicting the next token based on the previous ones. They are designed with a transformer architecture [71] with parameters ranging from 7 billion to 70 billion, and pretrained and fine-tuned on massive text datasets containing trillions of tokens collected from public sources [70]. Considering Llama 2 as the ML model f in Eq. (1), each instance in the training dataset



(a) Detection performance **(b) Encoder utility**

Figure 2: The impact of epochs on the detection performance and encoder utility. The evaluated encoder was trained by SimCLR on marked CIFAR-100 (10% are marked). The results are averaged over 20 experiments.

or fine-tuning dataset is a sequence of tokens (e.g., by a tokenizer defining a token vocabulary $\mathcal{V}$) of length L, i.e., $a_i = c_i^1 c_i^2 \dots c_i^L$ ($c_i^l \in \mathcal{V}$ for any $l \in \{1, 2, \dots, L\}$) and the loss function is defined as:

$$\ell(f, a_i) = \sum_{l=1}^{L} -\log \left[f(c_i^1 \dots c_i^{l-1}) \right]_{c_i^l}, \tag{4}$$

where $\left[f(c_i^1 \dots c_i^{l-1}) \right]_{c_i^l}$ denotes the c_i^l-th component of vector $f(c_i^1 \dots c_i^{l-1})$.

It is challenging to conduct lab-level experiments on auditing data in a pretrained Llama 2 because pretraining Llama 2 on a massive text corpus needs a huge amount of computing resources. Therefore, instead of applying our data auditing method to the pretrained Llama 2, we mainly focus on a Llama 2 fine-tuning setting.

6.2.1 Score Function. We used the negative loss, a simple and effective membership inference metric [8], as the score function. Formally, given a text sample x_i^β, we have $g^f(x_i^\beta) = -\ell(f, x_i^\beta)$, where $\ell(f, x_i^\beta)$ is defined in Eq. (4).

6.2.2 Experimental Setup.

Datasets. We used three text datasets: SST2 [62], AG's news [82], and TweetEval (emoji) [48]:

- **SST2**: SST2 is a dataset containing sentences used for sentiment analysis (i.e., there are 2 classes, "Negative" and "Positive"). In SST2, there are 67,300 training samples and 872 validation samples that we used for testing.
- **AG's news**: AG's news is a dataset containing sentences partitioned into 4 classes, "World", "Sports", "Business", and "Sci/Tech". In AG's news, there are 120,000 training samples and 7,600 test samples.
- **TweetEval (emoji)**: TweetEval (emoji) is a dataset containing sentences partitioned into 20 classes. In TweetEval (emoji), there are 100,000 training samples and 50,000 test samples.

Marking setting. In each experiment, we uniformly at random sampled a subset of training samples of a dataset as $\mathcal{X}$ (e.g., $|\mathcal{X}| =$ 10,000). From $\mathcal{X}$, we uniformly at random sampled $N = 1{,}000$ sentences $\{x_i\}_{i=1}^N$ assumed to be owned by a data owner. We applied our data marking algorithm to generate the published data $\{x_i^{b_i}\}_{i=1}^N$ and

the unpublished data $\{x_i^{1-b_i}\}_{i=1}^N$ for $\{x_i\}_{i=1}^N$. In Eq. (2), we defined the distance function by Levenshtein distance [34] and the utility difference function by semantic dissimilarity [18]. Instead of solving Eq. (2) exactly, we approximated it by using a paraphraser model (e.g., [72]) to generate two semantically similar but distinct sentences. We set $\hat{\mathcal{X}} = \{x_i^{b_i}\}_{i=1}^N$. As such, we constituted the training dataset collected by the ML practitioner as $\mathcal{D} = (\mathcal{X} \setminus \{x_i\}_{i=1}^N) \cup \hat{\mathcal{X}}$, labeled correctly (i.e., using their original labels).

Fine-tuning setting. We used `Llama-2-7b-chat-hf` Llama 2 model released in Hugging Face[4] as the base model. We used QLoRA [17] to fine-tune the Llama 2 model on the marked dataset, where we applied AdamW [42] as the optimizer that was also used to pretrain Llama 2 by Meta AI [70]. We fine-tuned the model with a learning rate of 2×10^{-4}. The fine-tuned Llama 2 is the ML model deployed by the ML practitioner.

Detection setting. In the detection algorithm, we set $\alpha = 0.025$ and $p = 0.05$.

Metrics. We used the following metrics to evaluate methods:

- **Test accuracy** (acc): acc is the fraction of test samples that were correctly classified by the fine-tuned Llama 2. A higher acc indicates a better performance of the fine-tuned model.
- **Detection success rate** (DSR): please see the description of this metric in Sec. 5.2.
- **Ratio between the number of queried published data and the total number of training samples** ($\frac{Q}{M}$): please see the description of this metric in Sec. 5.2.

6.2.3 Experimental Results. The results on applying our auditing method to the fine-tuned Llama 2 on three marked datasets are presented in Table 7. As shown in Table 7, when we tested the detection method on the pretrained Llama 2 (i.e., in the row of "Epoch 0"), we obtained a DSR of 0/20, indicating that the Llama 2 is not pretrained on the published data. If true, this result empirically confirms the bounded false-detection rate of our method. When we fine-tuned Llama 2 on the marked datasets by only 1 epoch, the accuracy of the fine-tuned Llama 2 model increased from 63.07% to 95.33% for SST2, from 28.41% to 91.69% for AG's news, and from 16.58% to 40.49% for TweetEval. At the same time, our method achieved a DSR of 20/20 on the fine-tuned model, which demonstrates the effectiveness of our method. Fine-tuning Llama 2 with more epochs increased the accuracy slightly (e.g., from 95.33% to 95.56% for SST2, from 91.69% to 92.33% for AG's news, and from 40.49% to 43.03% for TweetEval) but leads to a much lower $\frac{Q}{M}$. This is because fine-tuning the model for more epochs memorizes the fine-tuning samples more and the detection method needs fewer queries to the model to detect their use.

6.3 CLIP

In this section, we apply data auditing to a multimodal model [52, 54]. A multimodal model is a type of ML model that can understand and process various types of data, e.g., image, text, and audio. We considered Contrastive Language-Image Pretraining (CLIP) [52], developed by OpenAI in 2021, as our study case. CLIP is a vision-language model consisting of a visual encoder and a text encoder used to extract the features of the input image and text, respectively. It takes as inputs an image and a text and returns their corresponding feature vectors. CLIP is known for its notable performance in image-text similarity and zero-shot image classification [52]. As in Eq. (1), each instance is an image with its caption (i.e., a pair of image and text) and the loss function is the cross entropy loss used to push matched images and texts closer in the shared latent space while pushing unrelated pairs apart.

The CLIP model released by OpenAI was pretrained on 400 million image and text pairs collected from the Internet [52]. While it is challenging to pretrain such a large model on a huge number of pairs using lab-level computing resources, we aim to fine-tune the CLIP on a small (marked) dataset and test our auditing method on the fine-tuned CLIP.

6.3.1 Score Function. We defined the score function g^f by a recently proposed membership inference on CLIP [31]. It uses cosine similarity between the two feature vectors returned by the CLIP model as the inference metric [31]. Formally, given an image-text sample $x_i^\beta = (\dot{x}_i^\beta, \ddot{x}_i^\beta)$, we have $g^f(x_i^\beta) = \text{cosim}(f'(\dot{x}_i^\beta), f''(\ddot{x}_i^\beta))$, where f' and f'' are the visual encoder and text encoder of f, and cosim denotes cosine similarity.

6.3.2 Experimental Setup.

Datasets. We used the Flickr30k [80] dataset, which contains more than 31,000 images with captions. We used the first 25,000 as training samples and the remaining as test samples.

Marking setting. In each experiment, we uniformly at random sampled N images with captions from training samples $\mathcal{X}$. We set $\frac{N}{|\mathcal{X}|} = 10\%$, i.e., $N = 2{,}500$. We assumed these N captioned images are owned by a data owner. We applied our data marking algorithm to generate the published data $\{x_i^{b_i}\}_{i=1}^N$ and the unpublished data $\{x_i^{1-b_i}\}_{i=1}^N$ for $\{x_i\}_{i=1}^N$. In the marking algorithm, given a raw datum (i.e., an image with its caption), we followed the marking setting in Sec. 5.2 to generate two marked images and then randomly sampled one with its original caption as the published data, keeping the other as the unpublished data. We set $\hat{\mathcal{X}} = \{x_i^{b_i}\}_{i=1}^N$. As such, we constituted the training dataset collected by the ML practitioner as $\mathcal{D} = (\mathcal{X} \setminus \{x_i\}_{i=1}^N) \cup \hat{\mathcal{X}}$.

Fine-tuning setting. We used the CLIP model released by OpenAI[5] as the base model. We fine-tuned the CLIP model on the marked dataset $\mathcal{D}$, following the pretraining algorithm used by OpenAI [52]. We used a batch size of 256 and applied Adam [30] with a learning rate of 10^{-5} as the optimizer. The fine-tuned CLIP including the visual encoder and text encoder is the ML model deployed by the ML practitioner.

Detection setting. In the detection algorithm, we set $\alpha = 0.025$ and $p = 0.05$.

Metrics. We used the following metrics for evaluation:

- **Test accuracy** (acc): We randomly divided the test samples into batches (each is 256 at most). For each batch, we measured the fraction of texts correctly matched to images and the fraction of images correctly matched to texts, by the (fine-tuned) CLIP

[4] https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

[5] https://github.com/openai/CLIP

	SST2			AG's news			TweetEval (emoji)		
	acc%	DSR	$\frac{Q}{M}$	acc%	DSR	$\frac{Q}{M}$	acc%	DSR	$\frac{Q}{M}$
Epoch 0	63.07	0/20	10.00%	28.41	0/20	10.00%	16.58	0/20	10.00%
Epoch 1	95.33	20/20	2.87%	91.69	20/20	2.97%	40.49	20/20	3.89%
Epoch 2	95.26	20/20	0.22%	91.68	20/20	0.23%	41.88	20/20	0.26%
Epoch 3	95.56	20/20	0.12%	92.33	20/20	0.12%	43.03	20/20	0.12%

Table 7: Overall performance of our proposed method on Llama 2 fine-tuned on marked text datasets (10% of fine-tuning samples were marked) for different numbers of epochs, under an upper bound of $p = 0.05$ on the false-detection rate. All results were averaged over 20 experiments.

	acc%	DSR	$\frac{Q}{M}$
Epoch 0	80.73	0/20	10.00%
Epoch 1	88.44	20/20	6.99%
Epoch 2	88.53	20/20	2.31%
Epoch 3	88.53	20/20	1.21%

Table 8: Overall performance of our proposed method on CLIP fine-tuned on marked Flickr30k (10% of fine-tuning samples were marked) for different numbers of epochs, under an upper bound of $p = 0.05$ on the false-detection rate. All results were averaged over 20 experiments.

model. We used the fraction of correct matching averaged over batches as the test accuracy acc.

- **Detection success rate** (DSR): please see the description of this metric in Sec. 5.2.
- **Ratio between the number of queried published data and the total number of training samples** ($\frac{Q}{M}$): please see the description of this metric in Sec. 5.2.

6.3.3 Experimental Results. The overall performance of our data auditing method applied in fine-tuned CLIP is presented in Table 8. As shown in Table 8, when we audited the CLIP model released by OpenAI, we obtained a 0/20 DSR, which indicates that the pretrained CLIP model was not trained on our published data. If it is true that the CLIP model is not, this result empirically confirms the upper bound on the false-detection rate of our method. When we fine-tuned the CLIP model by the marked Flickr30k dataset, acc increased from 80.73% to 88.44% while DSR increased to 20/20, which demonstrates that our method is highly effective to detect the use of published data in the fine-tuned CLIP even when it is fine-tuned by only 1 epoch. When we fine-tuned the model for more epochs (e.g, 3 epochs), acc did not significantly increase. With more fine-tuning epochs, we still got a DSR of 20/20 but a smaller $\frac{Q}{M}$. Fine-tuning by more epochs made the model memorize the fine-tuning samples more and thus we needed fewer queries to the model in the detection step.

7 Discussion and Limitations

7.1 Minimal Number of Marked Data Required in Auditing

The minimal number of marked (published) data for which our method can detect its use depends on two factors: the memorization of training data by the ML model and the effectiveness of (contrastive) membership inference. For example, as shown in Sec. 5.3, the CIFAR-100 and TinyImagenet classifiers memorized their training samples more than the CIFAR-10 classifier, and so the data owner needed much less marked data to audit for data use in the CIFAR-100 and TinyImagenet classifiers than in the CIFAR-10 classifier. The effectiveness of (contrastive) membership inference also affects the minimal number of data items for which our method can detect use, i.e., a stronger membership inference method will allow our method to detect the use of fewer data. Therefore, we believe that any developed stronger membership inference methods in the future will benefit our technique.

7.2 Adaptive Attacks to Data Auditing Applied in Foundation Models

Once the ML practitioner realizes that the data auditing is being applied, he might utilize adaptive attacks aiming to defeat the auditing method when training his foundation models. Some adaptive attacks we considered for the image classifier (see Sec. 5.3.3) like early stopping, regularization, and differential privacy, can be used to mitigate the memorization of training/fine-tuning samples of foundation models. Therefore, these adaptive attacks could degrade the effectiveness or efficiency of our detection method. In addition, there are some methods used to mitigate membership inference in LLMs, e.g., model parameter quantization/rounding [50]. Any defense against membership inference in foundation models can be used as an adaptive attack. However, the application of these adaptive attacks will decrease the utility of the foundation models [50].

Since developers of foundation models usually aim to develop a powerful foundation model, they might hesitate to apply these adaptive methods since they will lose some model utility. As such, our data auditing method can pressure those developers of large foundation models to seek data-use authorization from the data owners before using their data.

7.3 Cost of Experiments on Foundation Models

In our experiments on auditing data use in foundation models (e.g., Llama 2 in Sec. 6.2 and CLIP in Sec. 6.3), we only considered model fine-tuning due to our limited computing resources. From the results shown in Sec. 6.2.3 and Sec. 6.3.3, our proposed method achieves good performance on detecting the use of data in fine-tuning Llama 2 and CLIP. We do believe that the effective detection performance of our method can be generalized to other types of foundation models and the settings where we audit the use of data in pretrained foundation models. This is because large foundation models memorize their training samples and thus are vulnerable to membership inference and other privacy attacks, as shown by existing works (e.g., [8, 31, 39, 50, 59]).

7.4 Toward Verifiable Machine Unlearning

One direct application of our data-auditing method is to verify machine unlearning. Machine unlearning is a class of methods that enable an ML model to forget some of its training samples upon the request of their owners. While there are recent efforts to develop machine unlearning algorithms [5], few focus on the verification of machine unlearning, i.e., verifying if the requested data has indeed been forgotten by the target model [63]. Our proposed method can be a good fit for verifying machine unlearning. Specifically, each data owner utilizes our marking algorithm to generate published data and hidden data. Upon the approval of data owners, a ML practitioner collects their published data and trains an ML model that can be verified by the data owner using our detection algorithm. If a data owner sends a request to the ML practitioner to delete her data from the ML model, the ML practitioner will utilize a machine unlearning algorithm to remove her data from his ML model and then inform the data owner of the successful removal. The data owner can utilize the detection algorithm to verify if the updated ML model still uses her published data. Our results in Sec. 5.3.1 show that our auditing method remains highly effective even when multiple data owners audit their data independently.

7.5 Proving a Claim of Data Use

Though our technique enables a data owner to determine whether an ML practitioner used her data without authorization, it alone does not suffice to enable the data owner to convince a third party. To convince a third party, the data owner should commit to $\{x_i^{b_i}\}_{i=1}^N$ and $\{x_i^{1-b_i}\}_{i=1}^N$ prior to publishing the former, e.g., by escrowing a cryptographic commitment to these data with the third party. Upon detecting use of her data by an ML practitioner, the data owner can open these commitments to enable the third party perform our hypothesis test on the ML model itself, for example. To enable a third party to replicate the data-owner's test result exactly, the data owner could provide the seed to a random number generator to drive the sequence of selections (WoR) from $\{I_1, \ldots, I_N\}$ in the test (see Sec. 4.2.1). However, to protect an ML practitioner from being framed by a malicious data owner, the data owner should be unable to freely choose this seed; e.g., it could be set to be a cryptographic hash of the commitments to $\{x_i^{b_i}\}_{i=1}^N$ and $\{x_i^{1-b_i}\}_{i=1}^N$.

8 Conclusion

In this paper, we proposed a general framework allowing a data owner to audit ML models for the use of her data. Our data auditing framework leverages any membership-inference technique, folding it into a sequential hypothesis test for which we can quantify the false-detection rate. Through evaluations of our proposed framework in the cases of an image classifier and various foundation models, we showed that it is effective, robust, and general across different types of ML models and settings. We thus believe our proposed framework provides a useful tool for data owners to audit ML models for the use of their data.

Acknowledgments

We thank the anonymous reviewers for their comments. This work was supported in part by NSF grants 2112562, 2125977, 1937787, and 2131859, as well as ARO grant No. W911NF2110182.

References

[1] [n. d.]. *AB-375 privacy: personal information: businesses.* https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375
[2] M. Abadi, A. Chu, I. Goodfellow, H B. McMahan, I. Mironov, K. Talwar, and L. Zhang. 2016. Deep learning with differential privacy. In *23rd ACM Conference on Computer and Communications Security.*
[3] S. Amari. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing* (1993), 185–196.
[4] S. Baluja. 2017. Hiding images in plain sight: Deep steganography. *30th Advances in Neural Information Processing Systems.*
[5] L. Bourtoule, V. Chandrasekaran, C. A Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot. 2021. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy.*
[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. 2020. Language models are few-shot learners. In *33rd Advances in Neural Information Processing Systems.*
[7] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. 2022. Membership inference attacks from first principles. In *43th IEEE Symposium on Security and Privacy.*
[8] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium.*
[9] F. Cayre, C. Fontaine, and T. Furon. 2005. Watermarking security: theory and practice. *IEEE Transactions on Signal Processing* 53 (2005), 3976–3987.
[10] S. Chatterjee. 2018. Learning and memorization. In *35th International Conference on Machine Learning.*
[11] M. Chen, Z. Zhang, T. Wang, M. Backes, and Y. Zhang. 2023. {FACE-AUDITOR}: Data auditing in facial recognition systems. In *32nd USENIX Security Symposium.*
[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *37th International Conference on Machine Learning.*
[13] C. A Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot. 2021. Label-only membership inference attacks. In *38th International Conference on Machine Learning.*
[14] I. N Cofone. 2020. *The right to be forgotten: A Canadian and comparative perspective.* Routledge.
[15] I. Cohen, Y. Huang, J. Chen, and J. Benesty. 2009. Pearson correlation coefficient. *Noise Reduction in Speech Processing* (2009).
[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition.*
[17] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. In *36th Advances in Neural Information Processing Systems.*
[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. (2019).
[19] L. Du, M. Chen, M. Sun, S. Ji, P. Cheng, J. Chen, and Z. Zhang. 2024. ORL-AUDITOR: Dataset auditing in offline deep reinforcement learning. In *31st ISOC Network and Distributed System Security Symposium.*
[20] C. Dwork. 2006. Differential privacy. In *33rd International Colloquium on Automata, Languages, and Programming.*
[21] P. Fernandez, G. Couairon, H. Jégou, M. Douze, and T. Furon. 2023. The stable signature: Rooting watermarks in latent diffusion models. In *IEEE International Conference on Computer Vision.*
[22] J. Geiping, L. Fowl, W R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein. 2021. Witches' brew: Industrial scale data poisoning via gradient matching. In *9th International Conference for Learning Representations.*
[23] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.
[24] J. Guo, Y. Li, L. Wang, S.-T. Xia, H. Huang, C. Liu, and B. Li. 2024. Domain watermark: Effective and harmless dataset copyright protection is closed at hand. In *38th Advances in Neural Information Processing Systems.*
[25] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition.*
[26] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference for Learning Representations.*

[27] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S Yu, and X. Zhang. 2022. Membership inference attacks on machine learning: A survey. *Comput. Surveys* 54 (2022), 1–37.

[28] Z. Huang, N. Z. Gong, and M. K. Reiter. 2024. A general framework for data-use auditing of ML models. *arXiv* 2407.15100 (Aug. 2024).

[29] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong. 2019. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *26th ACM Conference on Computer and Communications Security*.

[30] D. P Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*.

[31] M. Ko, M. Jin, C. Wang, and R. Jia. 2023. Practical membership inference attacks against large-scale multi-modal models: A pilot study. In *IEEE International Conference on Computer Vision*.

[32] A. Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Master's thesis. University of Toronto.

[33] Y. Le and X. S. Yang. 2015. Tiny ImageNet Visual Recognition Challenge. http://vision.stanford.edu/teaching/cs231n/reports/2015/pdfs/yle_project.pdf.

[34] V. I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, Vol. 10. Soviet Union, 707–710.

[35] Y. Li, Y. Bai, Y. Jiang, Y. Yang, S.-T. Xia, and B. Li. 2022. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. In *36th Advances in Neural Information Processing Systems*.

[36] Y. Li, M. Zhu, X. Yang, Y. Jiang, T. Wei, and S.-T. Xia. 2023. Black-box dataset ownership verification via backdoor watermarking. *IEEE Transactions on Information Forensics and Security* 18 (2023), 2318–2332.

[37] C.-J. Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Computation* (2007).

[38] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C L. Zitnick. 2014. Microsoft coco: Common objects in context. In *13th European Conference on Computer Vision*.

[39] H. Liu, J. Jia, W. Qu, and N. Z. Gong. 2021. EncoderMI: Membership inference against pre-trained encoders in contrastive learning. In *28th ACM Conference on Computer and Communications Security*.

[40] Y. Long, L. Wang, D. Bu, V. Bindschaedler, X. Wang, H. Tang, C. A Gunter, and K. Chen. 2020. A pragmatic approach to membership inferences on machine learning models. In *5th IEEE European Symposium on Security and Privacy*.

[41] I. Loshchilov and F. Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. In *5th International Conference for Learning Representations*.

[42] I. Loshchilov and F. Hutter. 2018. Fixing weight decay regularization in adam. In *6th International Conference for Learning Representations*.

[43] X. Luo, R. Zhan, H. Chang, F. Yang, and P. Milanfar. 2020. Distortion agnostic deep watermarking. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[44] A. Mantelero. 2013. The eu proposal for a general data protection regulation and the roots of the 'right to be forgotten'. *Computer Law & Security Review* (2013).

[45] A. Mao, M. Mohri, and Y. Zhong. 2023. Cross-entropy loss functions: Theoretical analysis and applications. In *40th International Conference on Machine Learning*.

[46] S. Merity, C. Xiong, J. Bradbury, and R. Socher. 2017. Pointer sentinel mixture models. In *5th International Conference for Learning Representations*.

[47] Y. Miao, M. Xue, C. Chen, L. Pan, J. Zhang, B. Z. H. Zhao, D. Kaafar, and Y. Xiang. 2021. The audio auditor: user-level membership inference in internet of things voice services. In *21st Privacy Enhancing Technologies Symposium*.

[48] S. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *International Workshop on Semantic Evaluation*.

[49] M. Nasr, R. Shokri, and A. Houmansadr. 2018. Machine learning with membership privacy using adversarial regularization. In *25th ACM Conference on Computer and Communications Security*.

[50] X. Pan, M. Zhang, S. Ji, and M. Yang. 2020. Privacy risks of general-purpose language models. In *41st IEEE Symposium on Security and Privacy*.

[51] B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *ACL Conference on Empirical Methods in Natural Language Processing*.

[52] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. A., G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *38th International Conference on Machine Learning*.

[53] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[54] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. 2021. Zero-shot text-to-image generation. In *38th International Conference on Machine Learning*.

[55] A. Sablayrolles, M. Douze, C. Schmid, and H. Jégou. 2020. Radioactive data: tracing through training. In *37th International Conference on Machine Learning*.

[56] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jégou. 2019. White-box vs black-box: Bayes optimal strategies for membership inference. In *36th International Conference on Machine Learning*.

[57] A. Saha, A. Subramanya, and H. Pirsiavash. 2020. Hidden trigger backdoor attacks. In *34th International Joint Conference on Artificial Intelligence*.

[58] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. 2019. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *26th ISOC Network and Distributed System Security Symposium*.

[59] W. Shi, A. Ajith, M. Xia, Y. Huang, D. Liu, T. Blevins, D. Chen, and L. Zettlemoyer. 2024. Detecting pretraining data from large language models. In *12th International Conference for Learning Representations*.

[60] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership inference attacks against machine learning models. In *38th IEEE Symposium on Security and Privacy*.

[61] K. Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference for Learning Representations*.

[62] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *ACL Conference on Empirical Methods in Natural Language Processing*.

[63] D. M. Sommer, L. Song, S. Wagh, and P. Mittal. 2022. Towards probabilistic verification of machine unlearning. In *Proceedings on Privacy Enhancing Technologies*.

[64] C. Song, T. Ristenpart, and V. Shmatikov. 2017. Machine learning models that remember too much. In *24th ACM Conference on Computer and Communications Security*.

[65] C. Song and V. Shmatikov. 2019. Auditing data provenance in text-generation models. In *25th ACM International Conference on Knowledge Discovery & Data Mining*.

[66] L. Song and P. Mittal. 2021. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX Security Symposium*.

[67] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *30th International Conference on Machine Learning*.

[68] M. Tancik, B. Mildenhall, and R. Ng. 2020. Stegastamp: Invisible hyperlinks in physical photographs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[69] R. Tang, Q. Feng, N. Liu, F. Yang, and X. Hu. 2023. Did you train on my dataset? towards public dataset protection with cleanlabel backdoor watermarking. *ACM SIGKDD Explorations Newsletter* (2023).

[70] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv* 2307.09288 (2023).

[71] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *30th Advances in Neural Information Processing Systems*.

[72] V. Vorobev and M. Kuznetsov. 2023. A paraphrasing model based on ChatGPT paraphrases. https://huggingface.co/humarin/chatgpt_paraphraser_on_T5_base.

[73] A. Wald. 1992. Sequential tests of statistical hypotheses. In *Breakthroughs in Statistics: Foundations and Basic Theory*.

[74] Z. Wang, C. Chen, L. Lyu, D. N Metaxas, and S. Ma. 2024. DIAGNOSIS: Detecting unauthorized data usages in text-to-image diffusion models. In *12th International Conference for Learning Representations*.

[75] I. Waudby-Smith and A. Ramdas. 2020. Confidence sequences for sampling without replacement. In *33rd Advances in Neural Information Processing Systems*.

[76] J. T.-Z. Wei, R. Y. Wang, and R. Jia. 2024. Proving membership in LLM pretraining data via data watermarks. *arXiv* 2402.10892 (2024).

[77] E. Wenger, X. Li, B. Y Zhao, and V. Shmatikov. 2024. Data isotopes for data provenance in DNNs. In *Privacy Enhancing Technologies Symposium*.

[78] J. Ye, A. Maddi, S. K. Murakonda, V.t Bindschaedler, and R. Shokri. 2022. Enhanced membership inference attacks against machine learning models. In *29th ACM Conference on Computer and Communications Security*.

[79] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *31st IEEE Computer Security Foundations Symposium*.

[80] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* (2014).

[81] N. Yu, V. Skripniuk, S. Abdelnabi, and M. Fritz. 2021. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *IEEE International Conference on Computer Vision*.

[82] X. Zhang, J. Zhao, and Y. LeCun. 2015. Character-level convolutional networks for text classification. In *29th Advances in Neural Information Processing Systems*.

[83] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu. 2019. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems* 30 (2019), 3212–3232.

[84] J. Zhu, R. Kaplan, J. Johnson, and F.-F. Li. 2018. Hidden: Hiding data with deep networks. In *European Conference on Computer Vision*.