RAI^2: Responsible Identity Audit Governing the Artificial Intelligence

Tian Dong*, Shaofeng Li†, Guoxing Chen*, Minhui Xue‡, Haojin Zhu* and Zhen Liu*
*Shanghai Jiao Tong University, China
†Pengcheng Laboratory, China
‡CSIRO's Data61, Australia

***Abstract*—Identity plays an important role in responsible artificial intelligence (AI): it acts as a unique marker for deep learning (DL) models and can be used to trace those accountable for irresponsible use of models. Consequently, effective DL identity audit is fundamental for building responsible AI. Besides models, training datasets determine what features a model can learn, and thus should be paid equal attention in identity audit. In this work, we propose the first practical scheme, named RAI^2, for responsible identity audit for both datasets and models. We develop our dataset and model similarity estimation methods that can work with black-box access to suspect models. The proposed methods can quantitatively determine the identity of datasets and models by estimating the similarity between the owner's and suspect's. Finally, we realize our responsible audit scheme based on the commitment scheme, enabling the owner to register datasets and models to a trusted third party (TTP) which is in charge of dataset and model regulation and forensics of copyright infringement. Extensive evaluation on 14 model architectures and 6 visual and textual datasets shows that our scheme can accurately identify the dataset and model with the proposed similarity estimation methods. We hope that our audit methodology will not only fill the gap in achieving identity arbitration but also ride on the wave of AI governance in this chaotic world.**

I. Introduction

Deep learning (DL) is reshaping our daily lives by performing complex real-world tasks such as image recognition [1], natural language processing (NLP) [2] and autonomous driving [3] thanks to large-scale datasets and models. Since the dataset collection (crawling and annotation) and large-scale deep neural networks (DNNs) model training requires huge labor and skill investment, DL properties (i.e., datasets and DNN models), especially high-quality ones with domain expert knowledge, are key *intellectual properties* (IPs) and are increasingly becoming the competitive advantages of tech giants. In knowledge-driven DL tasks (e.g., drug discovery [4, 5]), DL properties are even more important as they additionally relate to other IP (e.g., molecular obtained through costly biological experiments). Legislation like GDPR also prohibits and punishes unauthorized usage of DL properties containing personal data. Therefore, it is more than critical to protect the dataset and model from unauthorized duplication or inappropriate use [6] in responsible AI.

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.241012
www.ndss-symposium.org

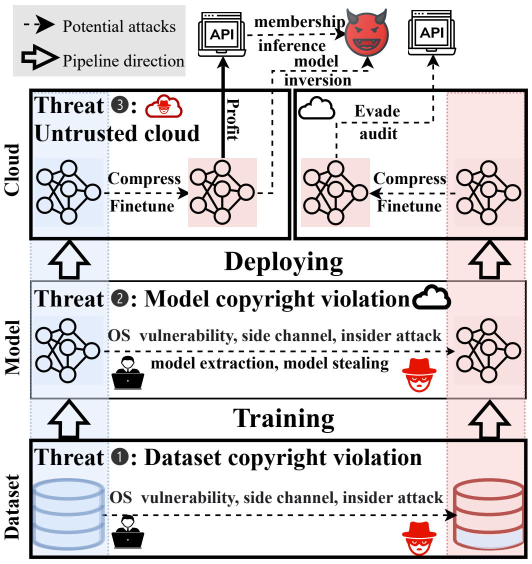
Potential attacks
Pipeline direction
API
membership inference
model inversion
API
Threat ❸: Untrusted cloud
Profit
Evade audit
Cloud
Compress
Finetune
Compress
Finetune
Deploying
Threat ❷: Model copyright violation
Model
OS vulnerability, side channel, insider attack
model extraction, model stealing
Training
Threat ❶: Dataset copyright violation
Dataset
OS vulnerability, side channel, insider attack

Fig. 1: Bottom-up IP threats existing in the DL pipeline. The threats for DL property originate from dataset-level (Threat ❶), model-level (Threat ❷) and untrustworthy platforms (Threat ❸). This work takes the first step of securing *both* dataset and model IPs via similarity estimation-based identity audit by a trusted third party.

Unfortunately, there is an increasing risk of copyright violation against datasets and models in the DL pipeline as shown in Fig. 1. During the training phase, valuable datasets are vulnerable to various operating system (OS) vulnerabilities and side-channel attacks [7]. Further, insider attack (e.g., data theft by departing employees) is regarded as a bigger threat than hacking. According to the survey by Biscom in 2021 [8], more than one in four respondents said they took data when leaving a company and 95% of respondents said that data theft was possible due to a lack of policies or technologies to prevent data stealing by leaving employees. This was confirmed by another independent survey by Tessian [9], which shows that 40% of US employees did take their generated data (or trained models) with them when leaving their job. In the deployment phase, extensive works are demonstrating the feasibility of unauthorized reproduction of DL models by leveraging the model stealing or extraction techniques [10, 11, 12]. These attacks pose severe threats to the DL intellectual property, calling for effective governance solutions in the sense of monitoring those possessions by identity audit.

Recently, a line of defense techniques has been proposed for DNN copyright protection, which mainly falls into two categories. The first one includes watermarking [13, 14, 15, 16, 17] that embeds a secret watermark (e.g., logo) into the protected model and fingerprinting [18, 19, 20, 21, 22] as a non-invasive alternative that extracts a unique fingerprint from the protected model. Both methods can proceed with black-box access to the suspect model. In particular, for the watermarking, the owner can query the suspect model by inputs containing a trigger to verify whether the suspect model has the specific watermark. For fingerprinting, the owner can verify the fingerprinting by querying the suspect model. Another line of work [22, 23, 24] compares the decision boundary similarity to protect model IP with black-box or white-box access to the suspect model. However, previous works are limited in various aspects: they fail to consider the training dataset and the judge practicability in IP disputes due to possibility of fraudulent ownership (see Sec. VI for more details).

Distinguished from the previous approaches, we propose a novel DL copyright protection framework by learning from the real-world software copyright protection methodology. According to the existing software copyright protection law, software copyright protection procedures can be simplified as follows: In the registration phase, the owner registered the software to a Trusted Third Party (TTP) (e.g., US Copyright Office) by submitting an electronic deposit. In the evaluation phase, a TTP evaluates its novelty by checking its originality. In the copyright infringement and remedies phase, an copyright infringer is liable for either the copyright owner's actual damages or any additional profits of the infringer.

However, several differences between the natures of DL property and software make it unsuitable to apply current software copyright protection for datasets and models. First, unlike software whose source code can be directly uploaded to a TTP and compared for similarity checking, enforcing dataset similarity and model similarity is not straightforward. For example, a slightly finetuned model contains totally different weights with the original one. Data augmentation techniques can render a dataset seemingly dissimilar. Second, the dataset and model are generally of larger size. A sample-by-sample or parameter-by-parameter comparison to estimate similarity is inefficient. Third, the infringer is reluctant to upload the dataset or model for privacy reasons, or can even cheat with an irrelevant dataset or model, thus requiring the TTP to undertake copyright infringement forensics with black-box similarity estimation methods.

These differences form several research challenges that should be addressed before constructing an AI copyright protection system, as we summarize as follows: *i)* Different from software copyright protection which can detect copyright infringement via source code scanning, how to determine AI copyright infringement represents the first challenge, especially when both dataset and model copyrights are jointly considered, given the potentially large size of the DL property. *ii)* How to achieve efficient and effective AI copyright registration to the TTP, and enable a TTP to discover copyright infringement through digital forensics without alerting the infringer, especially with only query access to the suspect model (a *black-box* manner) represents the second challenge.

We address the aforementioned challenges by proposing a novel responsible AI dataset and model audit scheme, RAI^2, which is comprised of two components: Similarity Estimation Module, and Third-Party Audit Module. The similarity estimation module (see Sec. III) is designed to detect copyright infringement on both AI dataset and model levels under black-box access to the suspect model. Note that the black-box access is necessary for practicability of secret audit by TTP. For the dataset, our intuition is that the confidence scores on trained data are significantly higher than those of unseen data due to DNN benign overfitting [25, 26, 27]. Therefore, we observe that dataset similarity is negatively correlated with the output difference between models trained on similar datasets: the more the overlap between the owner's and the suspect's datasets, the less the output difference between the two models trained on them. The proposed approach leverages this correlation to estimate the similarity between the protected and the suspect datasets, which can work accurately on a very small-size estimation set ($< 1\%$ of the protected dataset) rather than scanning the whole dataset. To measure two models' similarity with black-box model access, we introduce a novel metric of *model similarity* based on the distance between low-dimensional model projections obtained through querying models with uniformly distributed random inputs. The model level similarity checking is based on our observation that the model projection follows a distribution dependent on model weights, as validated by empirical evaluation with various mainstream DNN architectures, which can thus be exploited to measure weight similarity by comparing model projection distribution and to further detect model IP infringements.

To realize black-box IP infringement detection by forensics from a TTP, with similarity estimation methods, we propose a responsible audit scheme RAI^2 built upon a commitment scheme [28] and provide security proof of RAI^2. Our scheme allows a TTP to record registration of dataset or model identity and to determine both dataset and model identity of the audited part (suspect) in a black-box manner for real IP owner judgement. Taking both modules (similarity estimation and an audit scheme) together, we have a bedrock for IP regulation towards responsible AI. The main contributions of this work are summarized as follows:

- Inspired by the conventional software copyright protection, we devise the first DL identity audit scheme, termed RAI^2, which determines AI copyright infringement by jointly auditing dataset and model identity under black-box access to the suspect model.
- To accurately audit dataset and model identity, as a core component of RAI^2, we propose two novel metrics to estimate dataset and model similarities with black-box access to suspect models. Aided by the commitment scheme, our scheme enables the TTP to ascertain the actual owner with provable security guarantees by checking the IP registration order.
- We extensively evaluate two metrics on 6 visual and textual datasets and 14 DNN architectures. Our results validate the estimation accuracy of RAI^2. For example, on Tiny-ImageNet, our dataset similarity estimation has an error lower than 20% with 90% probability, and with the model similarity estimation, we can accurately classify finetuned models with 99.75% accuracy.

II. Responsible DL Identity audit

In this section, we present our motivation, the threat model and an overview of our identity audit scheme RAI^2.

A. Motivation

Similar as software, DL property (dataset and model) can be maliciously duplicated for illegal purpose. For example, suppose two competing companies work on object classification for autonomous driving service, or spam or toxic comment classification for cybersecurity service. One of them (called victim) invests huge resources to obtain high-quality dataset and model that brings competitive advantage. The other disadvantaged company (called adversary) aims to set up a service of equal quality using the same DL property, but legal copyright authorization is highly expensive and even unlikely to be feasible. Hence, the adversary seeks for unlicensed use or fraudulent ownership of the victim company's DL property. Moreover, the adversary can stealthily damage the deployed model to undermine the victim's competitiveness. Both cases violate the victim's IP. The threats come from any component of DL pipeline as summarized in the threat model (Sec. II-B).

To counter this, one solution is to check whether the suspect's property is similar enough to that of the victim: if the answer is affirmative, then it is a copyright violation, otherwise the suspect is innocent as she has independently created her DL property. Similarity checking requires the knowledge of what dataset or model is compared, thus we need to identify them first. As the ownership changes over time (e.g., by authorization), identifying them by the owner can be complex and confusing. We use the notion *identity* to represent the dataset samples (resp., model weights) for dataset (resp., model), and judge whether two datasets (or models) have the same identity by checking whether they are similar enough. The above similarity checking for DL dataset and model forms the basis of identity audit.

A naïve approach is to directly acquire and compare two datasets or models. However, such white-box access is not realistic as the large size makes it inefficient and the victim can also be concerned about potential leakage. On the other hand, the suspect may not hand out them or even can cheat the audit with a fake dataset and model. Therefore, it is indispensable to extract minimum but sufficient identity information (for efficiency) to enable similarity-based audit without being noticed by the adversary. In Sec. III, we describe how our similarity estimations extract the dataset and model identity information.

To ensure practicability, similarity alone is not enough: IP thief can also prove high similarity to victim's IPs using the proposed similarity estimation methods and accuse the victim of infringement, causing copyright dispute. Hence, our dataset and model similarity estimation methods need to be incorporated within current cryptographic tools to allow private or public verification. We show that, by incorporating with the cryptographic commitment scheme (e.g., [28]), it is possible to achieve TTP-based verification with our estimation methods, and propose our responsible *identity audit* scheme RAI^2. The TTP can be copyright administration or judicial authority, which allows the owner to register their IP to protect copyrights and is in charge of official ownership judgement by identity audit. Fig. 2 illustrates the process with RAI^2: the real owner first registers the property to a TPP, then it requests the TTP to audit the identity of suspect's property, and finally the TTP concludes whether the suspect has infringed the owner's IP or not. Next, we systematically categorize the IP threats in DL pipeline in the threat model.

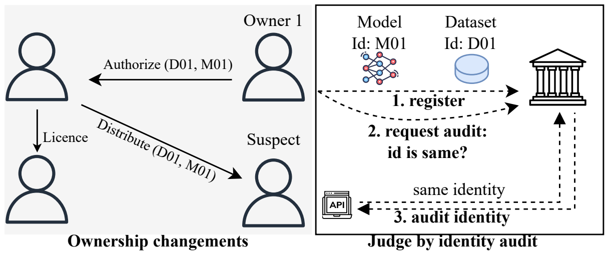

Fig. 2: Responsible use of AI requires dataset and DNN model IPs to be regulated. The ownership of DL property (dataset and model) can change (e.g., by legally trading or illegally stealing), but the identity remains unchanged. RAI^2 audits DL property identity to judge the infringement: the owner (indexed by 1) firstly registers properties (of identity M01 and D01) in advance to TTP that can later judge the ownership by black-box identity audit of the suspect's DL property.

B. Threat Model

The victim and the adversary (or suspect) are noted as $\mathcal{V}$ and $\mathcal{A}$, respectively. The dataset and model of $\mathcal{V}$ (or $\mathcal{A}$) are noted as $\mathcal{X}_{\mathcal{V}}$ (or $\mathcal{X}_{\mathcal{A}}$) and $f_{\mathcal{V}}$ (or $f_{\mathcal{A}}$). We assume both victim and adversary are interested in the common supervised classification task for competitiveness reason. The adversary targets for building a DL model of similar utility as that of the victim but does not know exact underlying distribution of victim data. The threats against victim's DL property $(\mathcal{X}_{\mathcal{V}}, f_{\mathcal{V}})$ in the DL pipeline (Fig. 1) are categorized as below:

Threat ❶: dataset copyright violation. As the model performance highly depends on training data quality, the adversary aims to build $\mathcal{X}_{\mathcal{A}}$ by stealing $\mathcal{X}_{\mathcal{V}}$ to train high-quality model. The adversary can exploit OS vulnerabilities or side-channel attacks [7] to steal $\mathcal{X}_{\mathcal{V}}$. Also, the adversary can be an insider attacker that directly obtains the dataset. For example, a former employee of lower rank in company $\mathcal{V}$ took away $\mathcal{V}_{\mathcal{X}}$ when leaving the job. Further, besides the dataset $\mathcal{X}_{\mathcal{V}}$ itself, we assume the adversary has no knowledge about the evidences (e.g., watermarks) generated for copyright protection for the dataset $\mathcal{X}_{\mathcal{V}}$ by $\mathcal{V}$. After stealing dataset $\mathcal{X}_{\mathcal{V}}$, the adversary's goal is to build her dataset $\mathcal{X}_{\mathcal{A}}$ composed by subset of $\mathcal{X}_{\mathcal{V}}$ and adversary's unrelated data of the similar distribution as $\mathcal{X}_{\mathcal{V}}$ to maintain data utility. Also, the adversary does not modify the samples of $\mathcal{X}_{\mathcal{V}}$ to preserve data utility. With $\mathcal{X}_{\mathcal{A}}$, the adversary trains a model $f_{\mathcal{A}}$ to achieve test accuracy close to $f_{\mathcal{V}}$.

Threat ❷: model copyright violation. In this category, the adversary aims to steal the model $f_{\mathcal{V}}$ directly. Particularly, the adversary can leverage OS vulnerabilities or side-channel attacks to steal the model $f_{\mathcal{V}}$, and she could also be an insider attacker (e.g., a former employee). Model weight extraction attacks to steal the model weights [12] also fall in this category. To evade potential copyright protection, the adversary tries to build $f_{\mathcal{A}}$ by deliberate modifications of model $f_{\mathcal{V}}$ with the cost

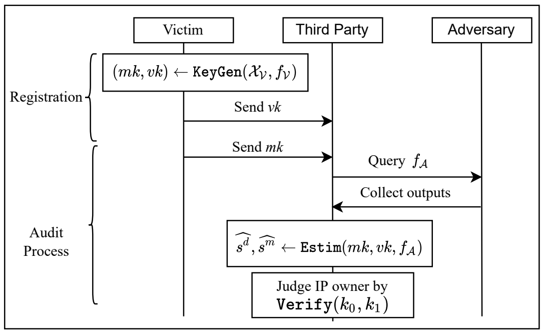
Victim
Third Party
Adversary
Registration
$(mk, vk) \leftarrow \texttt{KeyGen}(\mathcal{X}_\mathcal{V}, f_\mathcal{V})$
Send *vk*
Send *mk*
Query $f_\mathcal{A}$
Collect outputs
Audit Process
$\widehat{s^d}, \widehat{s^m} \leftarrow \texttt{Estim}(mk, vk, f_\mathcal{A})$
Judge IP owner by $\texttt{Verify}(k_0, k_1)$

Fig. 3: Illustration of identity *registration* to TTP and *audit* process in RAI^2.

that is significantly *lower* than training from scratch: model quantization, pruning or finetuning. Note that the finetuning is conducted on the adversary's own data of the similar distribution as $\mathcal{X}_\mathcal{V}$.

Threat ❸: untrusted server. Beside direct infringement, in this category, the adversary stealthily compresses the deployed model $f_\mathcal{V}$ to reduce the energy expenses at the cost of $f_\mathcal{V}$'s accuracy degradation, or to incriminates the victim in cases where, for example, the adversary (i.e., cloud provider) tries to harm the victim's reputation by causing bias or privacy issue in $f_\mathcal{V}$. The server stealthily modifies the victim's model by malicious finetuning (e.g., inducing unfairness or injecting trojans [29, 30]) with poisoned data of similar distribution, and accuses the victim when the modified model triggered lawsuits. Different from Threat ❷, the adversary does not target for model ownership but changes the model design without authorization.

Defense assumptions. The defender (i.e., TTP and victim) has *i)* no access to $\mathcal{X}_\mathcal{A}$, *ii)* black-box access to $f_\mathcal{A}$ by querying for confidence scores (in line with [31, 32]). Appendix D also provides evaluations with only top confidence scores.

Defense goals. The defender aims to *i)* detect dataset copyright violation by similarity estimation between $\mathcal{X}_\mathcal{V}$ and $\mathcal{X}_\mathcal{A}$, *ii)* detect the model copyright violation by similarity estimation between $f_\mathcal{V}$ and $f_\mathcal{A}$ and *iii)* limit the querying count (e.g., $\sim 10^3$) for efficiency.

C. Overview

We provide an overview of RAI^2. As conventional IP administration, we assume there exists a TTP in charge of DL property management who 1) records the DL property registration by identity and 2) arbitrates the real owner under request (in form of DL property identity audit). RAI^2 is designed for these objectives using three probabilistic polynomial-time (PPT) algorithms (`KeyGen`, `Estim`, `Verify`) that work between two parties indexed by 0 and 1:

- $\texttt{KeyGen}(\mathcal{X}_0, f_0)$ on input of dataset and model pair $\mathcal{X}, f_\mathcal{X}$ of party 0, generates key pair (mk_0, vk_0).
- $\texttt{Estim}(mk_0, vk_0, f_1)$ on input of key pair mk, vk and query access to f_1 of party 1, outputs $\widehat{s^d}, \widehat{s^m}$ as estimated dataset and model similarities between parties 0 and 1.
- $\texttt{Verify}(mk_0, vk_0, mk_1, vk_1)$ on input of key pairs from parties 0 and 1, outputs $b \in \{0, 1\}$ of the real owner index.

The victim can register dataset and model in advance to the TTP. For example, a company can register dataset to claim the ownership during patent application. The arbitrator can then utilize the committed values to audit the identity of the suspect's model and dataset via similarity estimation algorithm `Estim`. As the adversary can also register in the same way to imitate owner, the TTP needs `Verify` to judge the real owner. Fig. 3 illustrates the registration and audit phases:

Registration: The victim calls $\texttt{KeyGen}(\mathcal{X}_\mathcal{V}, f_\mathcal{V})$ to extract key pair (mk, vk) and register by verification key vk.

Identity Audit:

1. The third party estimates dataset and model similarity $\widehat{s^d}, \widehat{s^m} \leftarrow \texttt{Estim}(mk, vk, f_\mathcal{A})$ by querying $f_\mathcal{A}$ under request by victim with marking key mk provided by the victim.

2. The third party determines the real owner in copyright dispute between two parties (indexed by 0 and 1) by calling $\texttt{Verify}(mk_0, vk_0, mk_1, vk_1)$ to obtain output $b \in \{0, 1\}$ as judged owner party index.

We will present the construction of RAI^2 with security requirements and proof in Sec. IV. The audit is based upon the (estimated) dataset similarity $\widehat{s^d}$ and model similarity $\widehat{s^m}$ between the victim and the suspect to assess IP infringement. Therefore, it is necessary to build accurate similarity estimation methods, which is one of main contributions of this work. Before introducing the estimation, we first define the similarity.

D. Defining Similarity

Dataset similarity checking. The adversary steals victim's data for unauthorized use, so we define the *dataset similarity* of $\mathcal{X}_2$ relative to the base dataset $\mathcal{X}_1$ is

$$s^d_{\mathcal{X}_1}(\mathcal{X}_2) = |\mathcal{X}_1 \cap \mathcal{X}_2| \,/\, |\mathcal{X}_1| \in [0, 1], \tag{1}$$

where higher similarity indicates more data are stolen. Note that dataset similarity is not symmetric as it particularly quantifies the violation to the protected dataset instead of the suspect's dataset. Our experiments (Sec. V-C) show that our (estimated) similarity can accurately reflect the proportion of stealing data to victim for different size of $\mathcal{X}_\mathcal{A}$. To simply notation, we use the abbreviation s^d for $s^d_{\mathcal{X}_\mathcal{V}}(\mathcal{X}_\mathcal{A})$.

Model similarity checking. Direct weight comparison (e.g., by *p*-norm ($p \geq 1$) [23]) is only limited to continuous weight updates such as finetuning and does not cover other modifications. We propose to compare weight projections whose distribution is dependent on weight values. The projected distance between model f_1 and model f_2 is $D_w(\mathbf{h}_{f_1}, \mathbf{h}_{f_2})$ where $\mathbf{h}_{f_1}, \mathbf{h}_{f_2}$ are weight projections, D_w is a projection space metric, and the *model similarity* between two f_1 and f_2 is defined as

$$s^m_{f_1}(f_2) = \max(b_{D_w} - D_w(\mathbf{h}_{f_1}, \mathbf{h}_{f_2}), 0)/b_{D_w} \in [0, 1], \tag{2}$$

where b_{D_w} is a lower bound for distance between independently trained models that will be discussed in Sec. III-B. We use the abbreviation s^m for $s^m_{f_\mathcal{V}}(f_\mathcal{A})$ in the rest of the paper.

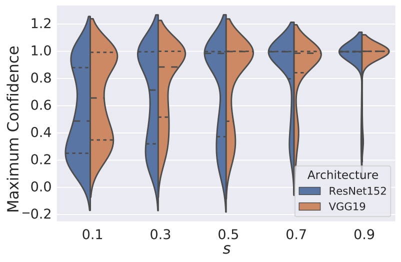

Fig. 4: Confidence distribution on $\mathcal{X}_2$ of model trained on $\mathcal{X}_1$ for different $s = s^d_{\mathcal{X}_1}(\mathcal{X}_2)$. Data are sampled from CIFAR-10.

III. Proposed Similarity Estimation

In this section, we show how to estimate the dataset and model similarity between the victim and the suspect.

A. Dataset Similarity Estimation

As for insights, in Fig. 4, the confidence distributions evaluated on $\mathcal{X}_2$ of models trained on $\mathcal{X}_1$ vary for different $s = s_{\mathcal{X}_1}(\mathcal{X}_2)$. We observe on different architectures that the distribution becomes more skewed to the maximum confidence as s^d increases. This one-to-one connection between dataset similarity and confidence (or output) distribution motivates us to estimate dataset similarity by model outputs. To estimate similarity, we propose to discretize the continuous range into several target similarity values (e.g., $\{0.0, 0.1, \cdots, 1.0\}$) and connect each target similarity to corresponding output distribution. The first step is to uniformly sample a small-sized estimation subset $\mathcal{S}_\mathcal{V} \subset \mathcal{X}_\mathcal{V}$. Then, we prepare a look-up table T_l which records the bijective correspondence between outputs on $\mathcal{S}_\mathcal{V}$ calibrated by $f_\mathcal{V}$ for different target similarity values. To estimate, it suffices to inquire the look-up table for $\widehat{s^d}$ whose corresponding output is the closest to $f_\mathcal{A}(\mathcal{S}_\mathcal{V})$ by statistical hypothesis testing. We quantify the closeness by *output distance* between two models f_1 and f_2 on dataset $\mathcal{X}$:

$$d_\mathcal{X}(f_1, f_2) = \mathbb{E}_{x\sim\mathcal{X}} \left\| f_1(x) - f_2(x) \right\|_2^2 . \tag{3}$$

Unless otherwise specified, the output distance refers to that between $f_\mathcal{V}$ and $f_\mathcal{A}$ on the estimation set $\mathcal{S}_\mathcal{V}$ used for query (i.e., $d_{\mathcal{S}_\mathcal{V}}(f_\mathcal{V}, f_\mathcal{A})$). Note that (3) can be generalized to p-norm because of norm equivalence in finite-dimensional vector space [33]. Proposition 1 enables us to estimate s^d based on output distance (proof in Appendix B).

Proposition 1. Suppose that the output distance on sample $x \in \mathcal{X}_\mathcal{V}$ is a random variable denoted by $X(x)$, and the conditional expectation of X on $\mathcal{X}_\mathcal{V}$ is strictly lower than that on other data, i.e., $\mathbb{E}[X(x)|x \in \mathcal{X}_\mathcal{V}] < \mathbb{E}[X(x)|x \notin \mathcal{X}_\mathcal{V}]$, then there is a linear correlation between output distance $d_{\mathcal{S}_\mathcal{V}}(f_\mathcal{V}, f_\mathcal{A})$ and the ground truth similarity s^d.

Our evaluation in Fig. 7 verifies that the correlation is approximately linear. Proposition 1 indicates that our look-up table-based method captures global model behavior on $\mathcal{S}_\mathcal{V}$ thus it can achieve more accurate estimation than sample-level estimation (e.g., by membership inference) as validated in Table II. Alg. 1 summarizes dataset similarity estimation in two parts: look-up table preparation (lines 1-5) and similarity estimation (lines 6-8), where the determination of $\widehat{s^d}$ is based on the average difference between two distributions measured by t-test statistics.

Algorithm 1: Estimating dataset similarity.

```
Input: $f_\mathcal{V}$, $f_\mathcal{A}$, target similarities $L_s$, estimation
       subset $\mathcal{S}_\mathcal{V}$.
Output: Estimator $\widehat{s^d}$.
1 $\mathrm{T}_l \leftarrow \{\}, L_{att} \leftarrow \emptyset$;
  /* Prepare the look-up table.                    */
2 for $s \in L_s$ do
3     Train surrogate $f^{sur}_s$ on $\mathcal{X}_s$ s.t. $s_{\mathcal{X}_\mathcal{V}}(\mathcal{X}_s) = s$;
4     $\mathrm{T}_l[s] \leftarrow \{\|f^{sur}_s(x) - f_\mathcal{V}(x)\|_2^2 \,| x \in \mathcal{S}_\mathcal{V}\}$;
5 end
  /* Compute $\widehat{s^d}$.                        */
6 $L_{att} \leftarrow \{\|f_\mathcal{A}(x) - f_\mathcal{V}(x)\|_2^2 \,| x \in \mathcal{S}_\mathcal{V}\}$;
7 Determine $\widehat{s^d}$ whose $\mathrm{T}_l[s]$ is closest to $L_{att}$;
8 return $\widehat{s^d}$
```

Improve efficiency. Training surrogate models f^{sur}_s is costly (Alg. 1 line 3). We propose an efficient heuristic that only needs two models $f^{sur}_{0.0}$ and $f^{sur}_{1.0}$. Specifically, outputs of f^{sur}_s on $\mathcal{S}_\mathcal{V}$ can be approximated by

$$f^{sur}_{0.0}(\mathcal{S}_\mathcal{V} \cap \mathcal{X}^\complement_\mathcal{V}) \cup f^{sur}_{1.0}(\mathcal{S}_\mathcal{V} \cap \mathcal{X}_\mathcal{V}), \tag{4}$$

where $f^{sur}_{0.0}$ is trained on disjoint dataset $\mathcal{X}^\complement_\mathcal{V}$ (e.g., public pretrained model). The heuristic reduces the complexity of surrogate model preparation from $O(|L_s|)$ to $O(1)$. Sec. V-C validates the effectiveness.

B. Model Similarity Estimation

Our method is inspired by random projection [34] which states that data projected by random matrix preserve distance in probability. We propose to project the weights with random inputs into low-dimensional data (called model projection) whose distribution depends on weight values and use the projections to measure model similarity. Fig. 5 compares random projection (left) and our approach (right).

Proposition 2. For random vector $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and linear layer of weight $\mathbf{W} \in \mathbb{R}^{1\times d}$ with bias $b \in \mathbb{R}$, the projection $Y = \mathbf{W}\mathbf{X} + b$ verifies $Y \sim \mathcal{N}(b, \|\mathbf{W}\|_2^2)$.

As an example, Proposition 2 (proof in Appendix B) shows how linear model weights determine the model projection distribution. For linear model, we obtain the analytic solution of model projection distribution and resort to Monte Carlo (MC) method for empirical computation (Fig. 6 (a)). As indicated by Fig. 6 (b), independent training results in less similar model projection distribution due to training randomness while finetuning leads to closer projections. Hence, in this case, we take $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ as the input and estimate $(b, \|\mathbf{W}\|_2^2)$ from the outputs. Then we use Euclidean distance between tuples $(b, \|\mathbf{W}\|_2^2)$ to measure model similarity.

The complexity of DNN architecture precludes the analytic solution of output distribution, thus we choose the autoencoder to learn the latent distribution and use the reconstruction error as the model distance (i.e., D_w in Sec. II-D), with which we can compute $\widehat{s^m}$ with reconstruction error bound b_{AE} for

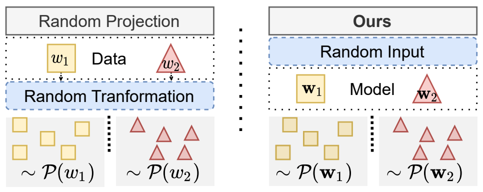

Fig. 5: Comparison between random projection and our model similarity estimation. The random projection exploits random transformations to project fixed data. Our method uses random inputs to project the fixed models weights.

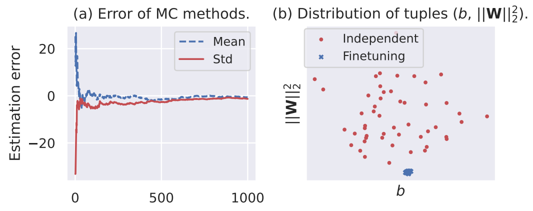

Fig. 6: (a) Verification of Proposition 2 with linear models trained on diabetes dataset of scikit-learn [35] and optimized by SGD. (b) The distribution of tuples $(b, \|\mathbf{W}\|_2^2)$ estimated through MC methods for 50 finetuned models (blue) and 50 independently trained models (red). Finetuned models share closely distributed $(b, \|\mathbf{W}\|_2^2)$ while that of independently trained models are scattered.

independent models. Alg. 2 shows how to estimate the model similarity. Specifically, the victim collects in advance model projection y_0 of his own model, trains an autoencoder for capturing latent distribution of y_0 and determines the error bound b_{AE} of independent models by, for instance, finetuning several epochs (lines 1-3). To estimate similarity, it suffices to query suspect model with n_{samp} random inputs and compute $\widehat{s^m}$ with the reconstruction error of the autoencoder (lines 4-7). The estimated similarity is then used for resolving potential model modification. Typically, except the *independent model* ($\widehat{s^m} \approx 0$), we are interested in the following two modifications:

- *Static modification* ($\widehat{s^m} \approx 1$): the suspect model shares the same weights as $f_{\mathcal{V}}$. In addition, model compression techniques (i.e., quantization or pruning) can be applied to speed up the inference.

- *Finetuning* ($0 < \widehat{s^m} < 1$): the suspect model is finetuned from $f_{\mathcal{V}}$ with data of similar distribution as $\mathcal{X}_{\mathcal{V}}$.

IV. Realization of RAI^2

Inspired by [13], we realize our proposed DL identity audit scheme. More detailed modeling and complete security proof can be found in Appendix C.

A. Realization

We first briefly overview the commitment scheme as our scheme is based upon it. Next, we realize our scheme based on dataset and model similarity estimations and commitment scheme. In the last, we provide security requirements.

Commitment scheme. We leverage the commitment scheme [28] to realize identity registration to the third party, because of its hiding and biding properties. The commitment scheme is a cryptographic primitive that allows the user to send some secret (i.e., committed value) to the receiver while keeping it confidential, and to unlock the secret to the receiver afterwards [28]. Formally, a commitment scheme is composed of the following two probabilistic polynomial time (PPT) algorithms:

- $\texttt{Com}(x, h)$: the commitment algorithm that encrypts the secret x and a random bit string h into c_x.

- $\texttt{Open}(c_x, x, h)$: the opening algorithm that returns 1 if $c_x = \texttt{Com}(x, h)$ and otherwise 0.

A commitment scheme should also verify that the committed value x in c_x cannot be changed after the commitment (*binding*) and that the receiver cannot distinguish $c_x \leftarrow \texttt{Com}(x, r)$ and $c_y \leftarrow \texttt{Com}(y, r)$ with any PPT algorithm for $x \neq y$ (*hiding*). If distributions of c_x and c_y are statistically close, the commitment scheme is *statistically hiding*.

Next, we realize our scheme based on dataset and model similarity estimations and commitment scheme, and provide security requirements.

Algorithm 2: Estimating model similarity.

```
  Input: $f_{\mathcal{V}}$, $f_{\mathcal{A}}$.
  Output: Estimator $\widehat{s^m}$.
  /* Prepare autoencoder.                         */
1 $y_0 \leftarrow f_{\mathcal{V}}(\text{rand}(n_{samp}))$ ;
2 $h_{AE} \leftarrow \text{TrainAutoecnoder}(y_0)$;
3 Determine bound $b_{AE}$ of $h_{AE}$ (e.g., by finetuning);
  /* Compute $\widehat{s^m}$.                      */
4 $\mathcal{S}_{\mathbf{h},\mathcal{A}} \leftarrow f_{\mathcal{A}}(\text{rand}(n_{samp}))$;
5 $e_{\mathcal{A}} \leftarrow \text{ReconstructError}(h_{AE}, \mathcal{S}_{\mathbf{h},\mathcal{A}})$;
6 $\widehat{s^m} \leftarrow \max(b_{AE} - e_{\mathcal{A}}, 0)/b_{AE}$;
7 return $\widehat{s^m}$.
```

$\texttt{KeyGen}(f_{\mathcal{V}}, \mathcal{X}_{\mathcal{V}})$:

1. Compute $V = (\mathcal{S}_{\mathcal{V}}, \mathcal{O}_{\mathcal{V}}, \mathrm{T}_l, h_{AE}, b_{AE})$, where $\mathcal{O}_{\mathcal{V}} = f_{\mathcal{V}}(\mathcal{S}_{\mathcal{V}})$ and t is the timestamp.
2. Sample random strings r_V to generate commitment $c_V \leftarrow \texttt{Com}(V, r_V)$.
3. Set $mk \leftarrow (V, r_V)$ and $vk \leftarrow c_V$. Return (mk, vk).

$\texttt{Estim}(mk, vk, f_{\mathcal{A}})$:

1. Parse $mk = (V, r_V)$ and $vk = (c_V)$. Check that $\texttt{Open}(c_V, V, r_V) = 1$. If not, return -1.
2. Obtain $\mathcal{O}_{\mathcal{A}}$ by querying $f_{\mathcal{A}}$ with $\mathcal{S}_{\mathcal{V}}$. Estimate $\widehat{s^d}$ with $\mathrm{T}_l, \mathcal{O}_{\mathcal{V}}$ and $\mathcal{O}_{\mathcal{A}}$ (Alg. 1) and $\widehat{s^m}$ with h_{AE}, b_{AE} (Alg. 2). Return ($\widehat{s^d}$, $\widehat{s^m}$).

$\texttt{Verify}(mk_0, vk_0, mk_1, vk_1)$:

1. Call $\texttt{Estim}$ for (mk_0, vk_0, f_1) and (mk_1, vk_1, f_0) to get outputs x, x'.
2. Check $x \neq -1$ and $x' \neq -1$ and positive estimated similarities; otherwise, return -1.
3. Get timestamps t and t' from mk and mk', respectively. If $t < t'$, return 0; otherwise, return 1.

Realization. Specifically, with the statistically hiding commitment scheme ($\texttt{Com}$, $\texttt{Open}$), the victim calls $\texttt{KeyGen}$ to commit the necessary components for estimation (V), where the timestamp of commitment t is also included. The output includes a secret marking key mk and corresponding verification key vk for identity registration (Fig. 3). When auditing the identity of a suspicious model for verifying potential IP infringement, the third party calls `Estim` to estimate dataset similarity and model similarity (used for classify model modifications). Finally, to recognize malicious registration of stolen dataset or model, `Verify` distinguishes the earlier registrant as the DL property owner. With binding and hiding properties of commitment, our scheme can be proved secure in terms of security requirements as defined below:

Security requirements. We suppose the audit scheme should verify the following requirements mentioned in [13]:

Non-trivial identity. The adversary cannot produce in advance key pair $(\widetilde{vk}, \widetilde{mk})$ that pretend to be arbitrary registered dataset or model even if she knows the estimation algorithm.

Unremovability. The adversary cannot change the dataset or model identity within time t much lower than that required for independent creation while preserving the utility, even if she knows the estimation algorithm. Here the utility means test accuracy for model, and for dataset it means test accuracy of models trained on it.

Unforgeability. Even if the adversary knows the key vk, she cannot convince the third party of owning a highly similar dataset or model.

Order-preserving Registration. In case where the adversary steals DL property (e.g., by insider attack), we require that the adversary cannot exploit the scheme to maliciously claim ownership of DL property registered by the real owner.

B. Security Analysis

Now we prove that our scheme verifies the above requirements based on the assumption of independently created datasets and models as stated below.

Independent creation assumption. We assume that if two datasets (resp., models) are independently created (resp., trained from scratch), then the similarity between the datasets (resp., models) is 0, otherwise the similarity is strictly positive. That is, two independently created datasets share no common sample and two models independently trained from scratch have projected distance higher than b_{D_w}.

As for dataset, the assumption holds for tasks where the dataset creation process is unlikely to be reproduced. For example, driving data generated by user are different every time of collection, the internal emails used for spam or toxic comment detection are different across enterprises, and the medical data (e.g., X-ray images) are different for each patient.

On the model side, it is observed that post-training modifications result in weight changes of small magnitude in order to preserve model performance [23], thus the assumption is valid as long as the adversary investigates small cost (e.g., several finetuning epochs) for modifying model. In addition, finetuning for many epochs can change the domain learned by the model. For example, a ResNet-18 originally trained on CIFAR-10 can achieve 95.47% test accuracy but after finetuning 20 epochs on STL-10 (dataset similar to CIFAR-10) the test accuracy drops to 68.04%, which damages the utility of model and is not in the interest of adversary.

Based on the independence creation assumption, we obtain the following theorem about the security property of our scheme.

Theorem IV.1. Under the assumption of existence of commitment scheme and independence creation, RAI^2, constructed by aforementioned algorithms (`KeyGen`, `Estim`, `Verify`), verifies the non-trivial identity, unremovability, unforgeability and order-preserving registration.

We briefly present the proof sketch here. The complete proof is deferred to Appendix C. To prove non-trivial identity, it suffices to notice that the adversary has a negligible probability of guessing an approximately exact estimation set and an exact model as those of the victim, because of independent creation assumptions. To prove unremovability, as we assume above that no algorithm can obtain DL property with time significantly smaller than the time necessary for independent creation. If the adversary can win the unremovability security game by some algorithm $\mathcal{A}$, the algorithm must not depend on the verification key because of statistically hiding property of commitment scheme, then she can obtain an independently created DL property with time t of $\mathcal{A}$, which contradicts the assumption. The unforgablity can be proved by reducing the problem of winning the game to breaking the commitment scheme, and the order-preserving registration clearly holds by algorithm `Verify`.

V. Experimental Evaluation

In this section, we evaluate our similarity estimation methods. In Sec. V-B, we evaluate the dataset similarity estimation and show that the factors (architecture, training epochs, learning rates, etc.) have little influence on performance. In Sec. V-C, we perform a case study on facial attribute dataset and further evaluate the heuristic and impact of dataset size as well as adaptive attacks (i.e., malicious modifying sample). In Sec. V-D, we show derivatives of model by different modifications can be accurately classified and evaluate our method against adversarial finetuning as an adaptive attack. Code for reproducing our results is available at https://github.com/chichidd/RAI2.

A. Experimental Setup

Datasets & models. We use image classification datasets CIFAR-10/100 [36] and Tiny-ImageNet [37], text classification dataset AG-News [38]. Two non-overlapped facial attribute classification datasets FairFace [39] and UTKFace [40] are used for cross-dataset evaluation. Appendix A contains more detailed dataset description. We consider the following seven state-of-the-art image classification architectures: ResNet [41], VGG [42], DenseNet [43], MobileNet-V2 [44], WideResnet [45], ResNext [46], RegNet [1]. For the text classification, we finetune pretrained models as the common practice in NLP community, and we evaluate on BERT [2] variants [47]: Tiny-BERT, Mini-BERT and Small-BERT.

Model training. We consider the similarities $s^d \in \{0.0, 0.1, \cdots, 0.9, 1.0\}$ in dataset estimation experiment, and

TABLE I: Test accuracy (%) for different architectures. Note that the test accuracy is slightly lower than benchmark because models are trained on half of training dataset and fewer training data degrades performance.

CIFAR-10		CIFAR-100			Tiny-ImageNet		
Architecture	**Top 1**	**Architecture**	**Top 1**	**Top 5**	**Architecture**	**Top 1**	**Top 5**
ResNet18	91.88 ± 0.15	ResNet34	69.14 ± 0.34	89.33 ± 0.22	ResNet152	42.83 ± 0.48	66.64 ± 0.38
ResNet34	92.23 ± 0.13	ResNet101	70.90 ± 0.56	90.86 ± 0.41	VGG19	51.12 ± 0.27	72.50 ± 0.36
ResNet50	91.36 ± 0.21	VGG16	64.31 ± 0.24	85.17 ± 0.20	DenseNet121	49.78 ± 0.35	72.88 ± 0.22
VGG13	90.41 ± 0.14	ResNext101	72.15 ± 0.29	91.44 ± 0.11	WideResNet101	40.33 ± 0.74	63.99 ± 0.87

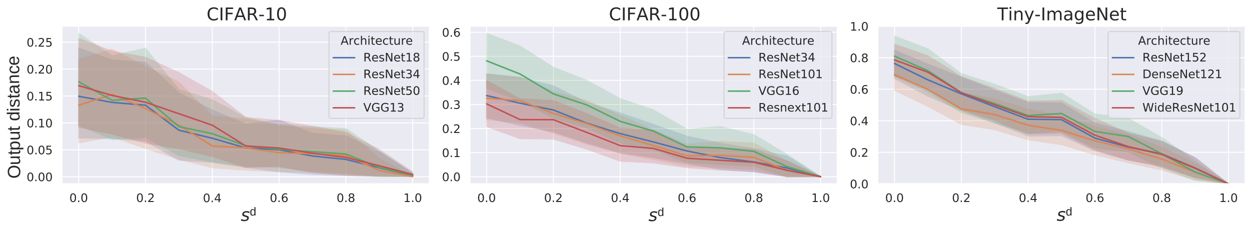

Fig. 7: Output distance across different values of dataset similarity s^d.

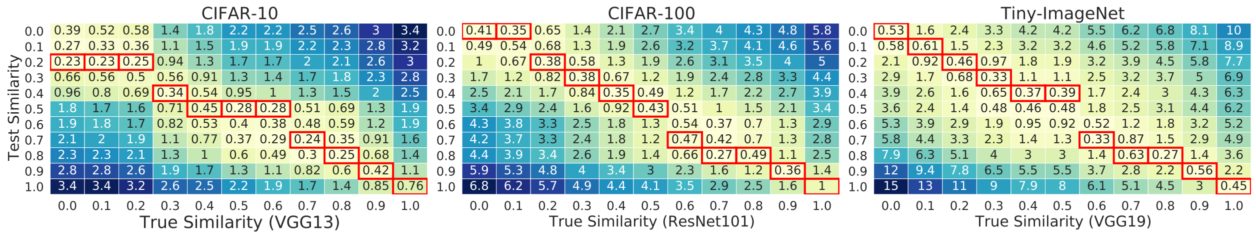

Fig. 8: Heatmap of statistics for dataset similarity estimation based on output distance.

repeat each training 10 times to reduce randomness and evaluate estimation accuracy. In total, we have trained more than 3,000 model copies for our experiments, which takes more than 15,000 GPU hours. Note that our experimental efforts do *not* imply impracticability of our approach. We adopt common techniques (dropout, weight decaying and data augmentations) to avoid overfitting (details in Appendix A1). Table I shows the averaged test accuracy scores for $f_{\mathcal{V}}$, which are slightly lower than training on whole datasets because of fewer training samples (e.g., only 25,000 samples for CIFAR-10). Small standard deviations indicate that our models have achieve stable performance. The test accuracy scores for finetuned NLP models are 91.78 ± 0.25 for Tiny-BERT, 92.11 ± 0.31 for Mini-BERT, and 92.37 ± 0.28 for Small-BERT.

B. Estimating Dataset Similarity

Dataset preparation. We first assume $\mathcal{A}$ and $\mathcal{V}$ have access to data of same distribution (*stronger* assumption than threat model), and use CIFAR-10/100, Tiny-ImageNet and AG-News for evaluation. To construct $\mathcal{X}_{\mathcal{V}}$ and $\mathcal{X}_{\mathcal{A}}$, we randomly split each dataset into two parts of equal size as the victim's dataset $\mathcal{X}_{\mathcal{V}}$ and independent dataset $\mathcal{X}_{\mathcal{V}}^{\complement}$, and pick s^d $(0 \leq s^d \leq 1)$ samples from $\mathcal{X}_{\mathcal{V}}$ and $1 - s^d$ samples from $\mathcal{X}_{\mathcal{V}}^{\complement}$ to form $\mathcal{X}_{\mathcal{A}}$.

Metric. In practice, we can only obtain an estimate $\hat{s}$ of the ground truth similarity. To quantify estimation error, we say an algorithm that produces $\hat{s}$ is (μ, ε)-*accurate* if

$$\Pr(|\widehat{s} - s| \geq \mu) \leq \varepsilon \tag{5}$$

for ground truth s. We adopt (μ, ε)-accuracy to evaluate our dataset similarity estimation. In our experiments, μ and ε are computed by repeating the estimation 10 times.

Victim models & Query subset. Since the architecture of $f_{\mathcal{V}}$ can be different from $f_{\mathcal{A}}$, without loss of generality, we set the victim's model architecture to ResNet18 for CIFAR-10, ResNet34 for CIFAR-100, ResNet152 for Tiny-ImageNet, and Tiny-BERT for AG-News (text classification task). We randomly select 100 samples as the query subset $\mathcal{S}_{\mathcal{V}}$ for image classification, and 600 samples for text classification, as in practice we found text models are less sensitive to individual input. Nevertheless, the proportion of query subset in the victim's training dataset is less than 0.5% in both tasks.

1) Linear relation between output distance and dataset similarity: Fig. 7 shows the output distances across different model architectures for each dataset similarity s^d. The bold line and the shaded band represent the average and the 95% confidence interval over 10 model copies, respectively. We observe that the output distance is approximately *linearly* dependent on the dataset similarity s^d, and the model architecture has little impact on the slope, because the curves locate closely to each other. This enables us to estimate dataset similarity s^d through the output distance. However, as shown in Fig. 9, the minor output distance differences caused by architecture can slightly affect the estimation accuracy, especially when the similarity is low. One solution is to use surrogate models of different architectures to improve the estimation accuracy.

Notice that the output distance of CIFAR-10 is lower than that of the other two datasets. We suppose this is because the number of classes of CIFAR-10 is much smaller. In other words, as the number of classes increases, the highest confidence score on non-training data samples should decrease, because the model is less confident when there are more choices. This renders the difference between model outputs on training and non-training data larger, and thus increases the output distance. We also observe that the largest output distance (i.e., case of $s^d = 0$) increases with the number of classes. Therefore, we suspect that, on large dataset (e.g., ImageNet), the slope should be larger, and as analyzed below, our output distance-based dataset similarity estimation will be more accurate.

2) Similarity Estimation: We illustrate how to evaluate $\widehat{s^d}$ in details. Following Alg. 1, the victim first prepares a look-up table T_l recording output distributions for different values of s^d, then compare them to the true output distribution of $f_{\mathcal{A}}$ to determine the similarity estimate $\widehat{s^d}$. The estimator $\widehat{s^d}$ is the s^d with smallest absolute value of *t*-test statistics.

In Fig. 8, we show the estimation results in the form of a heatmap and explain how to estimate. The horizontal axis is the ground truth dataset similarity s^d, and the vertical axis represents different similarity possibilities that the defender has tested. We also mark the architecture of the adversary's model $f_{\mathcal{A}}$ (unknown to the defender) in the label of the horizontal axis. Each column is computed as follows: for each ground truth s^d, we follow line 6 of Alg. 1 to generate L_{att} and follow line 7 of Alg. 1 to find the $\widehat{s^d}$ of closest distribution in look-up table T_l. We use *t*-test statistics as it measures the mean difference between two distributions. The value in each grid of the heatmap is the averaged absolute value of *t*-test statistics over 10 copies of $f_{\mathcal{A}}$. The minimal value of each column is highlighted with a red rectangle, whose test similarity becomes the estimated dataset similarity $\widehat{s^d}$. In this way, when testing dataset similarity on an unknown model, the defender collects the model outputs on $\mathcal{S}_{\mathcal{V}}$, applies line 6-7 of Alg. 1 to generate a column of statistics (like the heatmap column), and determine the estimator $\widehat{s^d}$ of minimal statistics.

For the most accurate estimation, the red rectangles should lie on the diagonal of the heatmap, because in this case, each ground truth dataset similarity (on the horizontal axis) is correctly estimated. However, in our experiment, we can observe that there is small deviation of one or two grids between the diagonal and the red rectangles' position, indicating the estimation error exists and can be quantitatively measured. Moreover, we can find that such deviation on the dataset CIFAR-10 is larger than that on the other two datasets. We suspect the cause is the variance of output distance: when the average output distance is close to the width of 95% confidence interval, e.g., the case of CIFAR-10 in Fig. 7, the estimation is less accurate, since *t*-test can be greatly influenced. On the other hand, when the average output distance is larger than the variance (e.g., the case of Tiny-ImageNet in Fig. 7), the estimation becomes more accurate. For CIFAR-100 and Tiny-ImageNet, only three red rectangles deviate one grid from the diagonal, indicating more accurate estimation.

3) Estimation Accuracy: We aim to quantify how accurate our estimation method is based on output distance. We propose to compute a probability bound ε for a given estimation error μ. The empirical results are shown in Fig. 9. In each figure, the bold line and the shaded area are the averaged ε and the 95% confidence interval computed over 10 model copies of $f_{\mathcal{V}}$ and $f_{\mathcal{A}}$. Since we have only evaluated $s^d \in \{0.0, \cdots, 1.0\}$, the error we evaluate here must be the multiple of 0.1. From Fig. 9, we can observe that the probability of an error larger than 0.1 is much higher than the probability of an error larger than 0.2. This is because the estimation error $\left|\widehat{s^d} - s^d\right|$ is 0.1 in most cases, i.e., $\Pr(\left|\widehat{s^d} - s^d\right| = 0.1) \gg \Pr(\left|\widehat{s^d} - s^d\right| > 0.1)$. It is also worth noting that the errors are all smaller than 0.4, which means that our estimation can distinguish three possibilities $s^d = 0$, $s^d = 0.5$ and $s^d = 1$ under the worst case. On larger dataset Tiny-ImageNet, this upper bound of the worst case decreases to 0.3. Therefore, we can claim that *our estimation can achieve an accuracy of* 0.1 *with high confidence, and can distinguish at least one* s^d *(*$0 < s^d < 1$*) under the worst case.*

Another observation is that *the architecture has slight impact on the estimation accuracy*. Take CIFAR-100 as an example, if the adversary trains a model $f_{\mathcal{A}}$ of architecture VGG16 on her own dataset $\mathcal{X}_{\mathcal{A}}$, the estimation error will be much larger than that of other similar architectures. However, as reported in Table I, VGG16 achieves the lowest accuracy score among all the architectures we evaluated (i.e., the adversary will sacrifice the model utility in this case), and thus VGG16 does not meet the adversary's expectation. In addition, we find that similar architectures result in more accurate estimation. This can be justified by the observation that the lowest curves occurring in the three figures in Fig. 9 all correspond to the victim's architecture.

We can also observe that our method achieves the best estimation accuracy on Tiny-ImageNet among the three tested datasets, i.e., the ε is lower given a certain μ. As mentioned above, it is due to Tiny-ImageNet has more classes. Thus, it becomes easier for our method to distinguish between cases of two adjacent s^d, e.g., case $s^d = 0.0$ and case $s^d = 0.1$. We leave the exploration on large datasets, e.g., ImageNet, as future work, due to the requirement of much more computational power.

Finally, it is important to point out that the (μ, ε)-estimation accuracy only describes the average accuracy across possible values of s^d and does not evaluate the estimation for a specific ground truth s^d. In Fig. 8, we can see that for s^d close to 1 (e.g., $s^d \geq 0.8$), the minimal statistics of the true similarity has larger difference wtih the rest test similarities, which indicates that *the defender can be more confident of estimation if the estimate* $\widehat{s^d}$ *is close to* 1. This also justifies the feasibility of further detecting model modifications if $\widehat{s^d} = 1$.

4) Sensitivity to the initial learning rate: We explore the impact of the initial learning rate used for the adversary's model training on the estimation accuracy. To simplify the presentation, we assume that $f_{\mathcal{V}}$ and $f_{\mathcal{A}}$ share the same architecture, and that the training epoch remains the same as the default setting, i.e., 100 for CIFAR-10 and 200 for the remainder. The learning rate decay also remains unchanged. In Fig. 10, we show the estimation accuracy for different initial learning rates across $\{0.01, 0.05, 0.1\}$, where 0.1 is the default learning rate. As expected, small learning rate increases the estimation error, because the model is not as well-trained as with large learning rate. Moreover, the influence of a small learning rate is amplified on larger datasets (i.e., CIFAR-100 and Tiny-ImageNet) because the hyperparameter

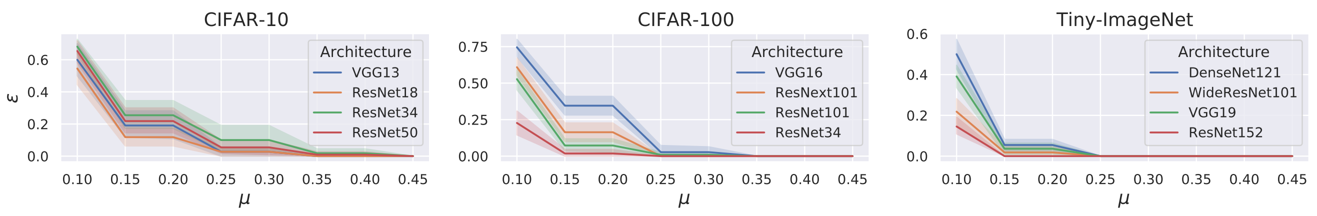

Fig. 9: (μ, ε)-estimation accuracy across different architectures.

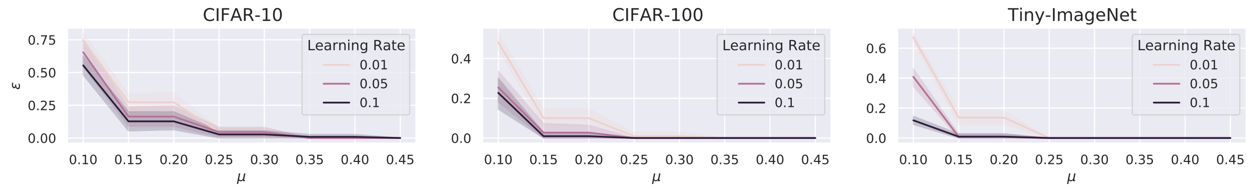

Fig. 10: Impact of learning rate of the adversary's model on estimation accuracy.

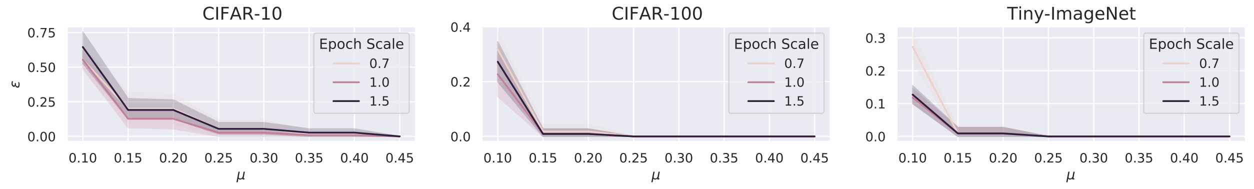

Fig. 11: Impact of training epochs of the adversary's model on estimation accuracy. The "epoch scale" is the ratio of the training epochs to the original one (i.e., 200 epochs).

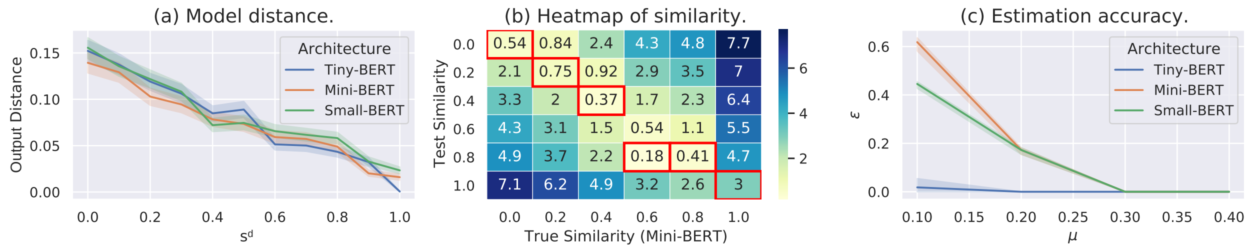

Fig. 12: Results of dataset similarity estimation on text classification dataset AG-News.

tuning is more important for training models on large datasets. Nevertheless, the models trained with the initial learning rate 0.01 on Tiny-ImageNet achieve the lowest test accuracy score (dropped nearly 25%), so it is unlikely for the attacker to choose small learning rates.

5) Sensitivity of training epochs: We also investigate the impact of training epochs of $f_{\mathcal{A}}$, with the default initial learning rate (i.e., 0.1). Suppose that the adversary trains $f_{\mathcal{A}}$ for n', and the default training epoch is n ($n = 100$ for CIFAR-10 and $n = 200$ for the remainder). Denote the epoch scale as the proportion $\frac{n'}{n}$. We explore the different scales $\{0.7, 1.0, 1.5\}$ and show the results in Fig. 11. Fewer training epochs increase the error for a larger dataset (Tiny-ImageNet), but the test accuracy is also lower than in the default case (i.e., the epoch scale is 1). As we can see, more training epochs will neither reduce or augment the estimation error.

Not that longer training with advanced mix-up techniques can enhance the model generalization and lead to larger estimation error. For example, using MixMo [48] and PreActResNet-18-2, it requires 1,000 training epochs to augment the test accuracy to 59.97 ± 0.99 while causing $\varepsilon = 0.63$ for $\mu = 0.2$. The reason is that well generalized adversary models are more confident (and accurate) on test data, making the estimator $\hat{s}$ higher than true s for small s. However, the adversary must acquire advanced training techniques and pay much more cost ($25\times$ more GPU hours to train one model with MixMo), which is not in the adversary's interest.

Textual dataset similarity. Similarly, we evaluate our method remains in NLP domain and show the main results in Fig. 12. In Fig. 12(a), we can see that the relationship between output distance and s^d is also linear with minor noises. We suspect this is because of repetition phrases in the news texts. Fig. 12(b) is the similarity heatmap between the victim's model (of architecture Tiny-BERT) and the adversary's model (of architecture Mini-BERT). The only deviated grid corresponds to $s^d = 0.6$. This is expected as the curve of $0.6 \leq s^d \leq 0.8$ in Fig. 12(a) is less steep, indicating that it is more difficult to distinguish a case for $0.6 \leq s^d \leq 0.8$. In Fig. 12(c), we observe that the language model's architecture also has influence on the estimation accuracy, especially when μ is small (e.g., $\mu = 0.1$). Nevertheless, we can observe that ε decreases sharply as μ increases, and becomes zero for $\mu = 0.3$, indicating that $s^d \in [\widehat{s^d} - 0.3, \widehat{s^d} + 0.3]$ for sure.

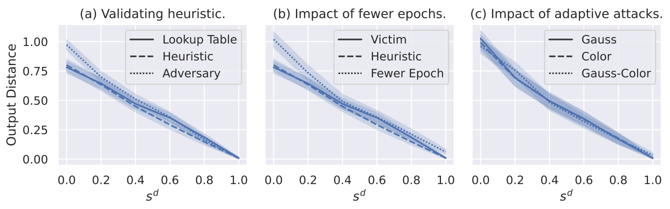

Fig. 13: Output distance for different s^d. (a) The heuristic can well approximate the ground-truth distance. Other factors including (b) training $f_{\mathcal{A}}$ with fewer epochs and (c) adaptively modifying samples have little impact on distance.

TABLE II: Comparison of estimation accuracy with membership inference attack (MIA). The best results are in bold.

	N.A.			Fewer Epoch		Gauss		Color		Gauss-Color	
μ	MIA	Heuristic	Ours	MIA	Ours	MIA	Ours	MIA	Ours	MIA	Ours
0.2	0.4	0.2	**0.1**	0.67	**0.2**	0.63	**0.17**	0.37	**0.1**	0.43	**0.2**
0.4	0.1	**0**	**0**	0.23	**0**	0.03	**0**	**0**	**0**	0.03	**0**

C. Case-study: Identify Facial Attribute Dataset

We have shown the estimation procedure and validated the effectiveness of our dataset similarity estimation. Next, we leverage the case-study on facial attribute classification to 1) verify the effectiveness for datasets of similar but different distributions (FairFace and UTKFace), 2) validate the heuristic (4) for reducing overhead, 3) evaluate against adaptive attacks and 4) investigate the impact of adversarial dataset size and anti-overfitting techniques. We regard FairFace as real-world, up-to-date dataset created by the victim and UTKFace as out-dated, less valuable dataset hold by adversary. Models trained on UTKFace have lower accuracy than those trained on FairFace on the test data from FairFace because of distribution shift, which motivates the adversary to steal data from the victim to improve the model quality.

Preparation. The datasets and models are prepared as follows: $\mathcal{X}_{\mathcal{V}}$ and $\mathcal{X}'_{\mathcal{V}} \subset \mathcal{X}^{\complement}_{\mathcal{V}}$ are two non-overlapped subsets randomly sampled from FairFace which are of same size as UTKFace, and $\mathcal{X}_{\mathcal{A}}$ is composed by mixing s^d data of $\mathcal{X}_{\mathcal{V}}$ and $1 - s^d$ of UTKFace for $s \in \{0.0, 0.2, \cdots, 0.8, 1.0\}$. The victim trains one ResNet101 $f_{\mathcal{V}}$ on $\mathcal{X}_{\mathcal{V}}$ and prepares the look-up table T_l using $\mathcal{X}_{\mathcal{V}}$ and $\mathcal{X}'_{\mathcal{V}}$. Note that the look-up table can be built for any sampled $\mathcal{X}'_{\mathcal{V}}$. The adversary trains more advanced RegNetY-8.0GF [1] on $\mathcal{X}_{\mathcal{A}}$ of $s^d \geq 0$ with $\mathcal{X}_{\mathcal{V}}$.

Fig. 13(a) presents output distance calculated from victim's look-up table, by heuristic and the ground-truth distance of adversary's RegNet trained on mixed dataset. Here, the surrogate model training is repeated on different sampled $\mathcal{X}^{\complement}_{\mathcal{V}}$ to build the look-up table. The heuristic curve (dashed) lies closely with the ground truth ones (solid and dotted), indicating our heuristic is effective for reducing overhead (5 times here) while retaining the utility. Fig. 13(b) shows that, even though the adversary sacrifices test accuracy (reducing from 49.05% to 44.95%) by training $f_{\mathcal{A}}$ with fewer epochs, the output distance remains close to the original one, which signifies that the similarity can still be accurately estimated.

Fig. 13 (c) shows the impact of three possible adaptive attacks (i.e., modifying training samples) on output distance: "Gauss" means blurring images with random Gaussian blur, "Color" means randomly changing the image's brightness, contrast, etc., and "Gauss-Color" means combination of "Gauss" and "Color". We consider these modifications as adaptive attacks against dataset similarity estimation because they directly change the image visual features (e.g., resolution, color). More details are in Appendix A2. With presence of adaptive attacks, the impact is little because curves in Fig. 13 (c) are close to the case without adaptive modifications (dotted curve in Fig. 13 (a)). This indicates our method is robust to sample-level modifications.

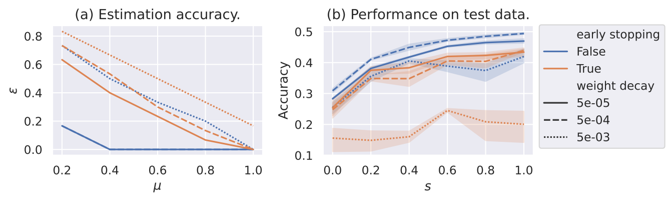

Fig. 14: Impact of early stopping and weight decaying on model performance and dataset similarity estimation accuracy.

TABLE III: Impact of adversary dataset size on estimation accuracy and comparison with baseline.

	$\lvert\mathcal{X}_{\mathcal{A}}\rvert$ / $\lvert\mathcal{X}_{\mathcal{V}}\rvert$							
μ	0.1		0.5		2		10	
	MIA	Ours	MIA	Ours	MIA	Ours	MIA	Ours
0.2	**0**	0.05	0.42	**0.37**	0.27	**0.05**	**0.52**	0.58
0.4	**0**	**0**	**0**	0.02	0.03	**0**	0.13	**0.02**

Table II compares the estimation accuracy (i.e., μ and ε) of our approach and Bayes optimal MIA [49]. The membership is inferred if the loss is lower than an optimal threshold. Specifically, we use the proportion of membership predicted by in $\mathcal{S}_{\mathcal{V}}$ as estimated similarity $\widehat{s^d}$, and round $\widehat{s^d}$ to the nearest true similarity value. We observe that for different μ and adaptive modifications, our approach obtains lower ε, thus is more accurate than MIA. Therefore, our approach systematically outperforms baseline MIA with or without adaptive modifications (i.e., attacks) to dataset.

Well-generalized adversary models. To further investigate how our approach performs if the adversary trains well-generalized model, we apply early stopping and study different weight decays in Fig. 14. Not surprisingly, applying early stopping and large weight decay increases dataset similarity estimation error but also reduces the model performance. Notably, early stopping causes larger estimation errors at expense of at least 11.1% accuracy drop comparing to the highest test accuracy achieved in our experiments with weight decay 5×10^{-4} and without early stopping. The results indicate that with existing techniques that prevent overfitting, the adversary cannot create larger estimation error to evade detection without sacrificing model performance.

Adversary dataset size. The adversary can adopt different dataset size. We increase the ratio $|\mathcal{X}_{\mathcal{A}}| / |\mathcal{X}_{\mathcal{V}}|$ by reducing the size of $\mathcal{X}_{\mathcal{V}}$ and decrease the ratio by reducing the size of $\mathcal{X}_{\mathcal{A}}$. Table III shows the estimation accuracy for different dataset size ratios and the comparison with MIA. We observe that when the if the adversary uses larger dataset, our approach can still achieve low estimation error and outperform the baseline.

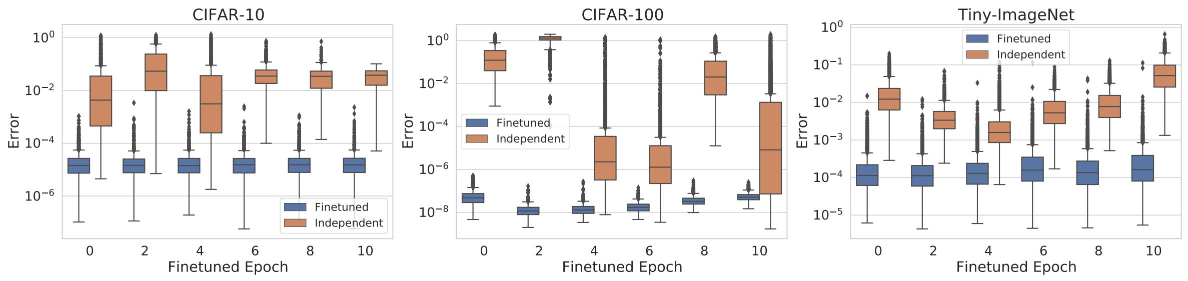

Fig. 15: Distribution of autoencoder's reconstruction error on finetuned models and sampled independently trained models.

Note that this case is in favor of adversary as more data improves model generalization, and the stolen data can only part of the whole adversary dataset. On the other hand, if the ratio is below than 1, the dataset similarity is bounded by the ratio as $s^d = \min(|\mathcal{X}_\mathcal{A}| / |\mathcal{X}_\mathcal{V}|, 1)$ by definition. In this case, our method generates larger error because fewer data degrades model generalization on non-training data, resulting in inaccurate output distribution matching. The poor model generalization does not affect the MIA, which is better than our method here. However, the smaller $\mathcal{X}_\mathcal{A}$ also greatly reduces the test accuracy from 49.05% to 40.02% for $|\mathcal{X}_\mathcal{A}| / |\mathcal{X}_\mathcal{V}| = 0.5$ and to 19.88% for $|\mathcal{X}_\mathcal{A}| / |\mathcal{X}_\mathcal{V}| = 0.1$. The damage to model utility prohibits the adversary to use smaller dataset.

D. Estimating Model Similarity

As a representative task of discrete inputs, NLP adopts pretrain-finetune paradigm, where the pretrained models are generally open-sourced and the dataset used for less costly finetuning [50] is more critical in NLP [51]. Therefore, in this work, we consider model of continuous inputs (e.g., image classification) as most DL tasks require continuous inputs, and leave the model of discrete inputs for future work.

Pre-processing. We train models of following architectures on $\mathcal{S}_\mathcal{V}$: ResNet18 for CIFAR-10, VGG16 for CIFAR-100 and MobileNet-V2 for Tiny-ImageNet. We set $n_{samp} = 1000$ to balance the trade-off between the performance and query costs. The inputs are sampled from the uniform distribution $\mathcal{U}(0, 1)$ and will be standardized into normalized inputs before being fed into network. For each model, we compute the model projection of length n_{samp}. Fig. 17 in Appendix A3 visualizes the model projection distribution via *t*-SNE.

Evaluation metrics. We evaluate the estimation as classification on critical similarity intervals as described in Sec. III-B: static modification ($\widehat{s^m} \approx 1$), finetuning ($0 < \widehat{s^m} < 1$) and independent model ($\widehat{s^m} \approx 0$). Thus, we use accuracy to globally measure the classification performance, and use False Positive rate and False Negative Rate as fine-grained metrics, where the negative represents independent model and the positive represents the other two modifications.

1) Static modifications: In this case, the reconstruction error is close to 0 so $\widehat{s^m} \approx 1$. Note that the null hypothesis of Kolmogorov–Smirnov (KS) test is "two samples are from the same distribution". As the static modification does not change weight values and is supposed to generate same distribution of model projection, we apply KS test on reconstruction errors for more accurate verification of $\widehat{s^m} \approx 1$. We provide the accuracy score (Top 1), KS test statistics and p-values for different statistic model transformations in Table IV. For quantization, we adopt static quantization to transform the model parameter into 8-bit. For pruning, we investigate the pruning rates equal to 20% and 40% to preserve the test accuracy. When p-value is low enough (e.g., lower than 0.01), the null hypothesis can be safely rejected. From Table IV, we can see that when the adversary applies no modification (i.e., $f_\mathcal{V} = f_\mathcal{A}$), model quantization, or model pruning, the *p*-values are higher than 0.01, and the defender can conclude that the adversary applied static modifications. Therefore, we can detect whether the adversary has applied static modification.

2) Finetuning & Adversarial Training: Beside vanilla finetuning, adversary can adopt adversarial training (AT) [52] as an adaptive countermeasure during finetuning to reduce similarity by larger weight updates during AT (as validated in Fig. 16). Hence, we consider both vanilla finetuning and finetuning with AT (called *adversarial finetuning*). We use the same finetuning setting for adversarial finetuning (i.e., applying adversarial training during finetuning): 20 epochs with a learning rate 1×10^{-4} for both normal and adversarial finetuning. As for the independent models, for each dataset, we train another 200 independent models using the same training hyperparameters but different initialization and random seeds. The independent models share different identities and are used for testing false positive rate.

Fig. 15 shows the reconstruction error distributions for the models finetuned less than 10 epochs, along with equal number of randomly selected independent models. We can see that there is a large gap between the distributions of independent models (orange) and finetuned models (blue). Moreover, the distributions of the finetuned model are generally stable and have little variation. Remark that the average of errors does not necessarily increase with the number of finetuned epochs. For example, in the middle of Fig. 15, the error average decreases at first and then increases. We suppose that this is because of the oscillations generated during model parameter optimization. Another possibility is that the autoencoder is not well trained (i.e., unable to well capture the distribution). As future work, we will explore other statistical testing methods for model projections (vectors).

According to Alg. 2, the finetuned and independent models are classified by a reconstruction error bound b_{AE}. We set b_{AE} as the maximum of averaged reconstruction error among 100 finetuned model copies. Specifically, the model is finetuned for 10 epochs, and we repeat the finetuning (both normal and adversarial) 5 times with different random seeds. As a result, we obtain $10 \times 5 \times 2 = 100$ model copies. Then, we use the threshold to classify 200 newly finetuned model copies

TABLE IV: Results of KS test for detecting static modifications (original, quantized, and pruned model). The *p*-values are all higher than 0.05, showing that the static transformation can be accurately detected.

Transformation	CIFAR-10			CIFAR-100			Tiny-ImageNet		
	Accuracy (%)	Statistic	*p*-value	Accuracy (%)	Statistic	*p*-value	Accuracy (%)	Statistic	*p*-value
Original	91.47	0.037	0.501	63.87	0.036	0.536	41.94	0.034	0.610
Quantized	91.55	0.035	0.573	63.87	0.021	0.981	41.28	0.062	0.042
Pruned (20%)	91.51	0.031	0.723	63.95	0.053	0.121	41.94	0.038	0.466
Pruned (40%)	91.40	0.042	0.341	63.95	0.030	0.759	41.94	0.037	0.501

TABLE V: Classification results of finetuned models and independent models.

Metric	CIFAR-10	CIFAR-100	Tiny-ImageNet
Accuracy (%)	96	94.25	99.75
False Positive Rate (%)	5	7	0
False Negative Rate (%)	3	4.5	0.5

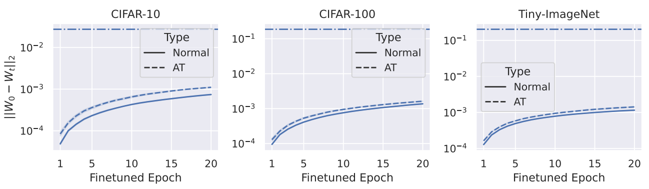

Fig. 16: Weight difference (L_2 norm) between normal and finetuned models. The dash-dotted horizontal line represents the minimum difference among 200 independently trained models. W_t is the weight of model finetuned t epochs.

that come from 5 times 20-epoch finetuning (both normal and adversarial), along with aforementioned 200 independent model copies.

Table V shows the classification results. Here, we use "False Positive" to denote the independent model copies misclassified as a finetuned model, and "False Negative" to represent finetuned model copies misclassified as an independent model. Their rates are the proportions among the number of ground truth (i.e., 200 for both classes). The results show that the threshold-based classification is able to achieve at least 94% accuracy score, and has at most 5% false negative (i.e., at most 5% finetuned model copies are classified as independent model copies). Note that, even though there are false positives, the independent model copies cannot be misclassified as the type "static transformations", as the maximum of *p*-values given by KS-test for independent model copies is less than 10^{-7}, demonstrating that our audit methodology can *at least* distinguish the static modification and independent models.

Ground truth weight difference. To better understand how finetuning affects ground truth weight difference, Fig. 16 shows that more finetuned epochs increases the L_2 norm of weight difference, and that independently trained models are of at least an order of magnitude higher difference than finetuned models, which validates our assumption in Sec. IV-B. Moreover, finetuning with AT is also shown to generate larger weight difference.

VI. Related Work

In this section, we review related work on DL identity (summarized in Table VI).

A. Model Identity

Existing works on identifying DL model fall into the following two categories: 1) watermarking and fingerprinting that hinge on unique model decision boundary feature as model marker; 2) similarity checking that identifies model by comparing whether the protected and suspect models are similar.

Watermarking & Fingerprinting. Model watermarking [13, 14, 15, 16, 17] protects model copyright through injecting into the model watermarks that are only known to the owner. Zhang et al. propose to embed watermarks similarly as the backdoor attacks [54] to protect model copyright. Adi et al. provide an effective cryptographic modeling for both watermarking and backdooring, allowing public ownership verifiability. Li et al. propose blind watermarks that are indistinguishable with normal samples to human eyes [15]. The entangled watermarking embedding [14] aims to preventing watermarking removal techniques such as model extraction [10]. Fingerprinting, as a non-invasive alternative, relies solely on the model's unique features (i.e., fingerprints) to identify models. Cao et al. propose the first model fingerprinting based on profiling the model decision boundary with test samples optimized to approach the boundary [18]. Lukas et al. and Wang et al. craft fingerprints that only lead the owner's and surrogate models (i.e., post-processed owner's model) to predefined outputs to examine [19, 20]. Recently, Peng et al. propose a fingerprinting method that could cope with model decision boundary modification [21] based on universal adversarial perturbation.

Similarity checking. Another line of research depends on model similarity to resolve identity: higher similarity between two models indicates that they are more likely to share the same identity. Chen et al. propose a testing framework for model copyright protection with white-box and black-box access to suspect models. The main approach is to measure the similarity between models using property-level metrics (black-box setting) or neuron-level metrics (white-box setting). Jia et al. quantify model similarity (in level of prediction) by approximating the decision boundary to linear models through LIME and computing Cosine distance between the linear models on reference points as similarity. However, these methods as well as fingerprinting are based on decision boundary, which is not dependable because the implicit decision boundary can be reproduced by independent training with specific DNN architectures [55]. On the other hand, Jia et al. propose Proof-of-Learning (PoL) [23] as a model weights verification mechanism based on proof-of-work, enabling the model ownership proof by repeating the model training process, but PoL requires white-box access to weights and cannot be applied to trained models. Our method compares weights under black-box access for similarity checking, because the weights are explicit and

TABLE VI: Related work on DL identity audit. Unlike existing approaches that hinge on model decision boundary and overlook dataset identity and fraudulent ownership claim, our method is the first *black-box* similarity checking for both model and dataset, implemented along with a commitment-based *practical* third-party audit scheme.

Approaches	**Model Identity**					**Dataset Identity**	**Practicability**	
	Method	Identification Basis	Evaluated Quantization	Evaluated Pruning	Evaluated Finetuning	Method	Access	Third-Party Audit
[18], [19],[20]	Fingerprinting	Decision Boundary	○	●	●	N.A.	Black-box	○
Peng et al. [21]	Fingerprinting	Decision Boundary	●	●	●	N.A.	Black-box	○
Maini et al. [53]	Inference	Training Data	○	●	●	Inference	Black-box	○
Jia et al. [23]	Proof-of-work	Model Weights	N.A.	N.A.	N.A.	N.A.	White-box	○
Chen et al. [22]	Similarity	Decision Boundary	○	●	●	N.A.	Black-box	○
Chen et al. [22]	Similarity	Neuron-level Behavior	○	●	●	N.A.	White-box	○
Jia et al. [24]	Similarity	Decision Boundary	●	●	●	N.A.	Black-box	○
RAI^2 (Ours)	Similarity	Model Weights	●	●	●	Similarity	Black-box	●

fixed, making it reliable for comparison. Moreover, the weights resemble to source code of computer programs which is copyrightable by laws [56], thus it is plausible to use model weights for model identification.

B. Dataset Identity

Similar as model watermarking, dataset watermarking [57, 58] has been proposed to protect the dataset copyright. To the best of our knowledge, no prior work has been proposed for identifying dataset in a non-intrusive way (i.e., without watermark). The most related work is on dataset and membership inference. Dataset inference [53] examines whether the suspect model is trained on owner's private dataset, thus can be used for dataset identity resolution. However, it cannot be applied if the private dataset is maliciously modified. For instance, the adversary can replace 10% dataset by data of similar distribution to prevent the dataset inference, yet most of samples remain same and the dataset identity should be judged unchanged. Membership inference attack (MIA) [31, 49] can also estimate dataset similarity, but it has been shown inaccurate by prior work [53] and our results.

VII. Conclusion and Discussion

In this work, we have proposed the first DL identity audit scheme, RAI^2. Our audit framework can effectively identify models and datasets based on similarity estimation. We have also realized our third-party audit framework through a provably secure commitment scheme, allowing IP registration to TTPs and infringement forensics. We hope that our methodology developed in this paper could facilitate responsible AI in this chaotic world. In the end, we discuss limitations and future research directions.

Technical aspects. We begin with the dataset. Estimating dataset similarity requires training surrogate models, which is remarkably expensive for large models and datasets. Nevertheless, in practice, AI developers are inclined to train multiple model copies (i.e., checkpoints) for model selections. These copies are thus available for dataset similarity estimation. Also, as we see in experiments, if the adversary trains a model of better generalization, the estimation error increases. Imagine an adversary trains a model that always predicts correctly with highest confidence, and it is impossible to infer s because there is no discrepancy between outputs on training and test data. In the future, we aim to improve our method's robustness to different level of generalization.

Further, we consider unauthorized dataset mixing as the primary dataset IP infringement, while there are more advanced dataset modification techniques such as InstaHide [59] and dataset condensation [60, 61, 62] applicable by adversary for regulatory compliance (e.g., privacy [59, 63]). Besides, some dataset can contain generated data (e.g., generated texts or images posted online) by web crawling [64] How to determine the IP rights and apply identity audit in these cases remains an open problem. Another direction is to make dataset unlearnable without permission key [65] or to develop the proof-of-creation (similar as PoL) that proves dataset ownership.

On the model part, our work only considers model weight as model IP, but exclude the model architecture. The adversary can only steal shallow layers that captures the visual details and retrain the rest, which also refers to a partial model IP infringement. To achieve covert audit, a more fine-grained black-box (thus practical) audit method for model identity is necessary. Also, privacy-enhancing techniques (e.g., differential privacy [66]) hides individual privacy but makes models indistinguishable between each other. Identity audit for privacy-preserving models (and their training data) is thus in urgent need. Last but not least, it is also important to explore a universal identity audit solution for different ML tasks.

Legal aspects. As model and dataset are electronic creation, the most related laws are those about software and database copyrights. To better solve disputes over ownership or transfer of rights, most countries provide a system allowing for the owners to register their copyrights [67]. When judging IP infringement, proving substantially similar play a central role even though there are multiple factors to be considered (e.g., access to the plaintiff's property) [56, 68]. RAI^2 offers a systematic approach to audit the identity registered in system based on our proposed similarity metrics to check substantial similarity for both model and dataset. What's more, RAI^2 is flexible, can be incorporated within existing software copyright registration systems to identify AI modules in protected programs. Other cryptographic tools (e.g., blockchains) are also helpful for building decentralized AI identity audit system. On the other hand, different from conventional programs and databases, the criteria of substantial similarity is still not clear and needs to be determined by actual law cases.

Acknowledgments

We thank anonymous reviewers for their fruitful comments. We further thank Hu Wang for discussion about random projection and Fangning Li for help from legal aspects. This work was supported by the National Key R&D Program of China (2022YFB3103500), National Natural Science Foundation of China under Grants, No. 62132013 and 61972453. Haojin Zhu and Minhui Xue are the corresponding authors of this paper.

References

[1] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proc. of IEEE/CVF CVPR*, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*, 2019.

[3] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[4] Tapping into the drug discovery potential of ai. https://www.nature.com/articles/d43747-021-00045-7, 2021.

[5] Wengong Jin, Jonathan M. Stokes, Richard T. Eastman, Zina Itkin, Alexey V. Zakharov, James J. Collins, Tommi S. Jaakkola, and Regina Barzilay. Deep learning identifies synergistic drug combinations for treating COVID-19. *Proc. Natl. Acad. Sci. USA*, 118(39):e2105070118, 2021.

[6] Danish Contractor, Daniel McDuff, Julia Katherine Haines, Jenny Lee, Christopher Hines, Brent Hecht, Nicholas Vincent, and Hanlin Li. Behavioral use licensing for responsible AI. In *Proc. of ACM FAccT*, 2022.

[7] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H. Lai. SgxPectre: Stealing intel secrets from sgx enclaves via speculative execution. In *Proc. of IEEE EuroS&P*, 2019.

[8] Biscom report. https://www.biscom.com/employee-departure-creates-gaping-security-hole-says-new-data/, 2021.

[9] Tessian report. https://www.tessian.com/blog/how-the-great-resignation-is-creating-more-security-challenges/, 2022.

[10] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *Proc. of USENIX Security*, 2016.

[11] Jean-Baptiste Truong, Pratyush Maini, Robert J. Walls, and Nicolas Papernot. Data-free model extraction. In *Proc. of IEEE CVPR*, 2021.

[12] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *Proc. of USENIX Security*, 2020.

[13] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proc. of USENIX Security*, 2018.

[14] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *Proc. of USENIX Security*, 2021.

[15] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN. In *Proc. of ACSAC*, 2019.

[16] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian M. Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proc. of ACM AsiaCCS*, 2018.

[17] Tianshuo Cong, Xinlei He, and Yang Zhang. Sslguard: A watermarking scheme for self-supervised learning pre-trained encoders. *arXiv preprint arXiv:2201.11692*, 2022.

[18] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proc. of ACM AsiaCCS*, 2021.

[19] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. In *ICLR*, 2021.

[20] Siyue Wang, Xiao Wang, Pin-Yu Chen, Pu Zhao, and Xue Lin. Characteristic examples: High-robustness, low-transferability fingerprinting of neural networks. In *Proc. of IJCAI*, 2021.

[21] Zirui Peng, Shaofeng Li, Guoxing Chen, Cheng Zhang, Haojin Zhu, and Minhui Xue. Fingerprinting deep neural networks globally via universal adversarial perturbations. In *Proc. of IEEE CVPR*, 2022.

[22] J. Chen, J. Wang, T. Peng, Y. Sun, P. Cheng, S. Ji, X. Ma, B. Li, and D. Song. Copy, right? a testing framework for copyright protection of deep learning models. In *Proc. of IEEE S&P*, 2022.

[23] Hengrui Jia, Mohammad Yaghini, Christopher A. Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *Proc. of IEEE S&P*, 2021.

[24] Hengrui Jia, Hongyu Chen, Jonas Guan, Ali Shahin Shamsabadi, and Nicolas Papernot. A zest of LIME: Towards architecture-independent model distances. In *ICLR*, 2022.

[25] Amartya Sanyal, Puneet K. Dokania, Varun Kanade, and Philip Torr. How benign is benign overfitting ? In *ICLR*, 2021.

[26] Niladri S. Chatterji and Philip M. Long. Finite-sample analysis of interpolating linear classifiers in the overparameterized regime. *Journal of Machine Learning Research*, 22(129):1–30, 2021.

[27] Peter L. Bartlett, Philip M. Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020. doi: 10.1073/pnas.1907378117.

[28] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proc. of ACM CCS*, 1999.

[29] Shaofeng Li, Minhui Xue, Benjamin Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[30] Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. Hidden backdoors in human-centric language models. In *ACM CCS*, 2021.

[31] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proc. of IEEE S&P*, 2017.

[32] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Proc. of NDSS*, 2019.

[33] norms on vector space. https://web.stanford.edu/class/math63cm/norms.pdf, 2019.

[34] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proc. of ACM SIGKDD*, 2001.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[36] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

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

[38] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proc. of NeurIPS*, 2015.

[39] Kimmo Kärkkäinen and Jungseock Joo. Fairface: Face attribute

dataset for balanced race, gender, and age for bias measurement and mitigation. In *IEEE WACV*, 2021.

[40] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proc. of IEEE CVPR*, 2017.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of IEEE CVPR*, 2016.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proc. of IEEE CVPR*, 2017.

[44] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

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

[46] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proc. of IEEE CVPR*, 2017.

[47] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: The impact of student initialization on knowledge distillation. *arXiv preprint arXiv:1908.08962*, 13, 2019.

[48] Alexandre Ramé, Rémy Sun, and Matthieu Cord. Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks. In *Proc. of IEEE/CVF ICCV*, 2021.

[49] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *Proc. of ICML*, 2019.

[50] Rongzhou Bao, Zhuosheng Zhang, and Hai Zhao. Span finetuning for pre-trained language models. In *Findings of EMNLP*, 2021.

[51] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proc. of ACL*, 2022.

[52] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[53] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *ICLR*, 2021.

[54] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *NIPS Workshop on Machine Learning and Computer Security*, 2017.

[55] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. In *Proc. of IEEE/CVF CVPR*, 2022.

[56] Bruce Perelman. Proving copyright infringement of computer software: An analytical framework. *Loy. LAL Rev.*, 18:919, 1985.

[57] Yiming Li, Ziqi Zhang, Jiawang Bai, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. Open-sourced dataset protection via backdoor watermarking. In *NeurIPS Workshop*, 2020.

[58] Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. In *NeurIPS*, 2022.

[59] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. InstaHide: Instance-hiding schemes for private distributed learning. In *Proc. of ICML*, 2020.

[60] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *ICLR*, 2021.

[61] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *Proc. of ICML*, 2021.

[62] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *CoRR*, abs/2110.04181, 2021.

[63] Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *Proc. of ICML*, 2022.

[64] Shaofeng Li, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Suguo Du, and Haojin Zhu. Backdoors against natural language processing: A review. *IEEE Security & Privacy*, 20(05):50–59, 2022.

[65] Mingfu Xue, Yinghao Wu, Yushu Zhang, Jian Wang, and Weiqiang Liu. Protect the intellectual property of dataset against unauthorized use. *arXiv preprint arXiv:2109.07921*, 2021.

[66] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proc. of ACM CCS*, 2016.

[67] Wipo copyright. https://www.wipo.int/copyright/en/, 2022.

[68] Mark A Lemley. Our bizarre system for proving copyright infringement. *J. Copyright Soc'y USA*, 57:719, 2009.

APPENDIX

A. Experimental Details

• **CIFAR-10/100 [36]:** The CIFAR-10 dataset is composed of 60,000 colored images of size 32×32 separated into 10 classes, where 50,000 samples are for training the rest 10,000 samples are for test. CIFAR-100 is similar as CIFAR-10, except that it has 100 classes and has disjoint images.

• **Tiny-ImageNet:** Tiny-ImageNet [37] is a subset of ImageNet containing 110,000 colored images of size 64×64 coming from 200 classes, among which 100,000 images are used for training and the remaining 10,000 for test.

• **AG-News [38]:** This dataset of text classification contains 120,000 news texts for training and 7,600 for test. The texts are grouped into 4 classes.

• **FairFace [39] & UTKFace [40]:** FairFace and UTKFace contain facial images of attribute classification. In particular, FairFace contains 7 race labels and 9 age groups. On the other hand, UTKFace only contains 5 race labels and the age values labeled from 0 to 116. We unify the labels and obtain 20 different attribute classes. The images are center cropped and resized to 128×128. Moreover, UTKFace only contains 23,705 images which is fewer than FairFace, so we use a randomly sampled FairFace training subset to simulate $s = 1$. We adopt the official test dataset (10,954 images) of FairFace for model performance evaluation.

1) Model training: For image data, we train each model for 100 epochs for CIFAR-10, FairFace and UTKFace, and 200 epochs for CIFAR-100 and Tiny-ImageNet. We set batch size to 128, and use SGD optimizer with momentum 0.9 and weight decay 5×10^{-4}. We The initial learning rate is 0.1, and is decayed by 0.2 on each decaying epoch. For CIFAR-10, the decaying epochs are 40 and 70. For the other two datasets, the decaying epochs are 60, 120 and 160. Additionally, we adopt various input diversifying techniques to alleviate overfitting, including random cropping, random rotation and random horizontal flipping. For text classification task, we finetune pretrained BERT models with batch size 64 and learning rate 5×10^{-4} for three epochs to achieve the highest accuracy on validation data.

2) Adaptive attack details for dataset similarity estimation: We tested two adaptive attacks against our dataset similarity estimation method. The first is blurring image with randomly chosen Gaussian blur of kernel size is 15. The minimal and maximal σ for blurring is 0.1 and 2.0 respectively. We implement with `GaussianBlur` of torchvision. The second is randomly changing the brightness, contrast, saturation and hue of image. We realize this with `ColorJitter` of torchvision and set 0.2 for all parameters.

3) Visualization of model projection: To be more illustrative, we provide *t*-SNE visualization of model projection (i.e., y_0 in line 1 of Alg. 2) in Fig. 17. We consider the following modifications that can be applied by the adversary: finetuning, pruning, quantization. Furthermore, we also take into account the other checkpoint files during the same training process (noted as "Checkpoint"), because they share similar model parameters with the original model (noted as "Origin"). The independent models are noted as "Independent". Seen from Fig. 17, we observe that the model projection of independent models are separated from that of associated models (i.e., models of type other than "Independent"), indicating that they are separable in high-dimensional space. In addition, the associated models' outputs are closely grouped, signifying that they share similar distribution in high-dimensional space. This also explains why we adopt autoencoder to capture the latent output distribution and use the reconstruction error to estimate the model similarity (i.e., small error means high similarity and vice versa).

B. Proofs

Proofs of Proposition 1. Let $B_{tr}(x) = \mathbb{1}_{x\in\mathcal{X}_\mathcal{A}}$ denote a random variable indicating whether a sample x belongs to the adversary's training dataset, where the $\mathbb{1}_A$ is the indicator of event A. Since the probability that a sample of $\mathcal{S}_\mathcal{V}$ belongs to $\mathcal{X}_\mathcal{A}$ is s, we have $B_{tr} \sim Bernoulli(s)$ on $\mathcal{S}_\mathcal{V}$ and $d_{\mathcal{S}_\mathcal{V}}(f_\mathcal{V}, f_\mathcal{A}) = \mathbb{E}_{x\sim\mathcal{S}_\mathcal{V}}[B_{tr}(x)X_{tr}(x)] = s\mathbb{E}_{x\sim\mathcal{S}_\mathcal{V}}[X(x)|x \in \mathcal{X}_\mathcal{A}] + (1-s)\mathbb{E}_{x\sim\mathcal{S}_\mathcal{V}}[X(x)|x \notin \mathcal{X}_\mathcal{A}]$. Hence, higher s leads to a smaller model distance on $\mathcal{S}_\mathcal{V}$, and the slope depends on $\mathbb{E}[X(x)|x \in \mathcal{X}_\mathcal{A}]$ and $\mathbb{E}[X(x)|x \notin \mathcal{X}_\mathcal{A}]$.

Proof of Proposition 2. Since $\mathbf{X} = [X_1, \cdots, X_d] \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, where $\mathbf{I}_d$ is the identity matrix, then for linear layer weight $\mathbf{W} \in \mathbb{R}^{1\times d}$, we have $\mathbf{WX} \sim \mathcal{N}(\mathbf{0}, \mathbf{W}^\top\mathbf{W})$, where $\mathbf{W}^\top\mathbf{W} = \|\mathbf{W}\|_2^2$. Moreover, for $Y = \mathbf{WX} + b$, the expectation of Y verifies $\mathbb{E}[Y] = b$. Hence, we obtain $Y \sim \mathcal{N}(b, \|\mathbf{W}\|_2^2)$.

C. Security Requirements and Analysis

1) Defining Identity: Now we provide basic modeling algorithms for dataset and model creation, and then define the identity for dataset and model.

Dataset creation. We assume private datasets of same domains are subsets a sufficiently large data pool $\mathcal{D} \subset \{0,1\}^*$ s.t. $|\mathcal{D}| = \Theta(2^{n_s})$, where $|\mathcal{D}|$ denotes the set $\mathcal{D}$ size and the $n_s \in \mathbb{N}$ is security parameter as implicit input for all algorithms. Note here $\{0,1\}^*$ can be seen as space of float-point numbers for instance. The dataset creation is modeled as $\mathcal{X} \leftarrow \texttt{GenDataset}(\mathcal{O}^d)$, where `GenDataset` is a PPT algorithm. We also assume that sampled dataset is equipped with ground-truth labels.

Model creation. Similarly, because of training randomness (e.g., by hardware), we regard training on dataset $\mathcal{X}$ as $f_\mathcal{X} \leftarrow \texttt{Train}(\mathcal{X})$, where `Train` is PPT algorithm, $f_\mathcal{X}$ is the trained model of weight $\mathbf{w}(f_\mathcal{X}) \subset \{0,1\}^{\text{poly}(n_s)}$. Let `Predict` be the deterministic polynomial-time algorithm that returns model f's confidence scores (within $[0,1]$) on dataset $\mathcal{S}$ in form of the vector set: $\mathcal{C}_{f,\mathcal{S}} \leftarrow \texttt{Predict}(f, \mathcal{S})$. DNNs are reported to overfit training data to lower generalization risk [25, 26, 27], we say (`Train`, `Predict`) is *α-fitted* on training dataset $\mathcal{S}$ if $\min_{v\in\mathcal{C}_{f_\mathcal{S},\mathcal{S}}} \max v \geq \alpha$ (i.e., α as lower bound of confidence on training data), and define the confidence margin between training and test data as $\beta = \alpha - \max_{v\in\mathcal{C}_{f_\mathcal{S},\mathcal{D}\setminus\mathcal{S}}} \max v$, which verifies $\beta \gg 1-\alpha$ due to overfitting.

To simplify notation, we define the DL property (i.e., dataset and model) creation as algorithm: `DLProperty`():

1. Generate $\mathcal{X} \leftarrow \texttt{GenDataset}(\mathcal{O}^d, z_\mathcal{X})$, $f_\mathcal{X} \leftarrow \texttt{Train}(\mathcal{X}, z_{f_\mathcal{X}})$.
2. Return $\mathcal{X}, f_\mathcal{X}$.

To check whether two datasets (or models) have same identity, we can only compare whether they are similar enough based on independent creation assumption. Next, we formalize for dataset and model separately.

Independent creation assumption. Original datasets are supposed to be significantly different between each other comparing to those obtained through copyright violation. Specifically, two independently created private datasets $\mathcal{X}_1, \mathcal{X}_2$ are expected to have no overlap with other dataset (e.g., driving data from two autonomous vehicle companies), so we assume the probability of existence of common sample is negligible in n_s. As for models, comparing to stolen and modified models (e.g., by finetuning), models trained from scratch (i.e., independently trained) should contain totally different weights because of training randomness (e.g., from SGD), resulting in lower similarity. Hence, we assume there exists an upper distance bound b_{D_w} s.t. for any two independently trained models f_1, f_2, the probability of their distance lower than b_{D_w} is negligible in n_s.

Requirements. Formally, the security requirements of RAI^2 can be formalized into games as follows:

Non-trivial identity. The adversary cannot produce in advance key pair $(\widehat{vk}, \widehat{mk})$ that pretend to be arbitrary registered dataset or model even if she knows the estimation algorithm. That is, the adversary has negligible chance of winning the following game with her PPT algorithm $\mathcal{A}()$:

1. Produce key pair $(mk_\mathcal{A}, vk_\mathcal{A}) \leftarrow \mathcal{A}()$.
2. Generate $\mathcal{X}_\mathcal{V}, f_\mathcal{V} \leftarrow \texttt{DLProperty}()$.
3. $(mk_\mathcal{V}, vk_\mathcal{V}) \leftarrow \texttt{KeyGen}(\mathcal{X}_\mathcal{V}, f_\mathcal{V})$.
4. Estimate $\widehat{s^d}, \widehat{s^m} \leftarrow \texttt{Estim}(vk_\mathcal{A}, mk_\mathcal{A}, f_\mathcal{V})$.
5. $\mathcal{A}$ wins if $\max(\widehat{s^d}, \widehat{s^m}) \approx 1$.

Unremovability. The adversary cannot change the dataset or model identity within time t much lower than that required for independent creation while preserving the utility, even if she knows the estimation algorithm. Here the utility means test accuracy for model, and for dataset it means test accuracy of models trained on it. Namely, the chance of winning the following game by adversary PPT algorithm $\mathcal{A}(vk_\mathcal{V}, f_\mathcal{V}, \mathcal{X}_\mathcal{V})$ is negligible.

1. Generate $\mathcal{X}_\mathcal{V}, f_\mathcal{V} \leftarrow \texttt{DLProperty}()$.
2. $(mk_\mathcal{V}, vk_\mathcal{V}) \leftarrow \texttt{KeyGen}(\mathcal{X}_\mathcal{V}, f_\mathcal{V})$.
3. Run $f_\mathcal{A} \leftarrow \mathcal{A}(vk_\mathcal{V}, f_\mathcal{V}, \mathcal{X}_\mathcal{V})$.
4. Estimate $\widehat{s^d}, \widehat{s^m} \leftarrow \texttt{Estim}(vk_\mathcal{V}, mk_\mathcal{V}, f_\mathcal{A})$.
5. $\mathcal{A}$ wins if $\min(\widehat{s^d}, \widehat{s^m}) \approx 0$ while preserving test accuracy, i.e.,

$$\Pr_{(X,y)\sim\mathcal{D}}[f_\mathcal{V}(X) = y] \approx \Pr_{(X,y)\sim\mathcal{D}}[f_\mathcal{A}(X) = y]. \tag{6}$$

Unforgability. Even if the adversary knows the key vk, she cannot convince the third party of owning a highly similar dataset or model. Formally, $\mathcal{A}$ can only win the following game using PPT algorithm $\mathcal{A}()$ with negligible probability:

1. Generate $\mathcal{X}_\mathcal{V}, f_\mathcal{V} \leftarrow \texttt{DLProperty}()$.
2. $(mk_\mathcal{V}, vk_\mathcal{V}) \leftarrow \texttt{KeyGen}(f_\mathcal{V}, \mathcal{X}_\mathcal{V})$.
3. Run $f_\mathcal{A}, \mathcal{X}_\mathcal{A}, mk_\mathcal{A} \leftarrow \mathcal{A}(vk_\mathcal{V}, f_\mathcal{V}, \mathcal{X}_\mathcal{V})$.
4. Estimate $\widehat{s^d}, \widehat{s^m} \leftarrow \texttt{Estim}(vk_\mathcal{V}, mk_\mathcal{A}, f_\mathcal{A})$.
5. $\mathcal{A}$ wins if $\max(\widehat{s^d}, \widehat{s^m}) \approx 1$ while verifying (6).

Proof of Theorem IV.1 Now we show the proof of Theorem IV.1. *Non-trivial identity.* To win the game, the adversary needs to achieve

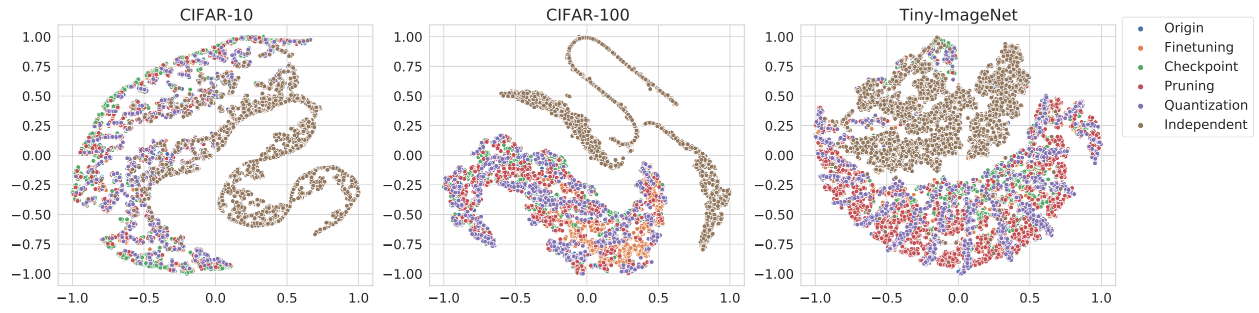

Fig. 17: t-SNE visualization of model outputs for different transformations.

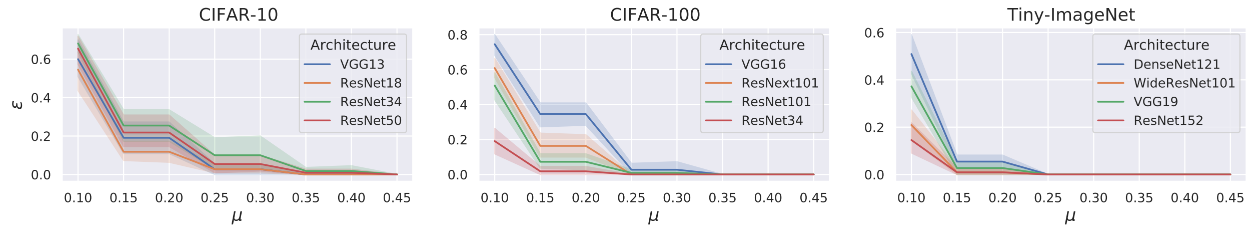

Fig. 18: (μ, ε)-estimation accuracy with only 5 largest confidence scores available.

TABLE VII: Results of KS test for detecting static modifications (original, quantized, and pruned model) with topest confidence scores.

Transformation	CIFAR-100		Tiny-ImageNet	
	Statistic	p-value	Statistic	p-value
Original	0.031	0.723	0.052	0.133
Quantized	0.034	0.610	0.033	0.648
Pruned (20%)	0.038	0.401	0.031	0.723
Pruned (40%)	0.040	0.466	0.029	0.795

TABLE VIII: Classification results of finetuned models and independent models with topest confidence scores.

Metric	CIFAR-100	Tiny-ImageNet
Accuracy (%)	94.25	100
False Positive Rate (%)	7	0
False Negative Rate (%)	4.5	0

$\widehat{s^d} \approx 1$ or $\widehat{s^m} \approx 1$ with constructed key pair $(\widetilde{vk}, \widetilde{mk})$. To achieve $\widehat{s^d} \approx 1$, the adversary needs to construct $\mathcal{S}_\mathcal{A}$ by sampling from $\mathcal{D}$ such that only few elements differ between $\mathcal{S}_\mathcal{A}$ and $\mathcal{S}_\mathcal{V}$, under assumption benefiting adversary that $f_\mathcal{A}$ is obtained by same α-fitted algorithms (`Train`, `Predict`). Denote $|\mathcal{D}| = n$ and the size of datasets $|\mathcal{S}_\mathcal{V}| = |\mathcal{S}_\mathcal{A}| = n_1$. The probability of εn_1 different elements is $\Pr[|\mathcal{S}_\mathcal{A} \setminus \mathcal{S}_\mathcal{V}| = \varepsilon n_1] < \binom{n_1}{n_1 - \varepsilon n_1}\binom{n}{\varepsilon n_1}/\binom{n}{n_1} = (\frac{n_1!}{(\varepsilon n_1)!})^2 \frac{(n-n_1)!}{(n_1-\varepsilon n_1)!(n-\varepsilon n_1)!}$, which is bounded by $\frac{n_1^{2n_1}}{(n-n_1)^{n_1-\varepsilon n_1}}$ that is around 0 for small εn_1 and increases with εn_1. As wining game with $\widehat{s^d} \approx 1$ is equivalent to minimum confidence error $(1-\alpha)n_1$ between $\mathcal{O}_\mathcal{A} \leftarrow \texttt{Predict}(f_\mathcal{A}, \mathcal{S}_\mathcal{A})$ and $\mathcal{O}_\mathcal{V} \leftarrow \texttt{Predict}(f_\mathcal{V}, \mathcal{S}_\mathcal{A})$, εn_1 different elements increase the error to $(1-\alpha)(n_1 - \varepsilon n_1) + \beta \varepsilon n_1$, where $\beta \gg 1-\alpha$ is the confidence margin between training and test data. According to Alg. 1, $\widehat{s^d} < 1$ as the output distribution does not match on εn_1 elements sampled by adversary. On the other hand, it is of negligible probability that two independently trained models (i.e., $f_\mathcal{A}$ and $f_\mathcal{V}$) achieve $\widehat{s^m} \neq 0$ because of the assumption on independent creation. Therefore, the adversary cannot win the game with non-negligible probability.

Unremovability. Assume that there exists no algorithm that can produce dataset or model of same utility (i.e., test accuracy) as that of $\mathcal{V}$ in time t significantly smaller than the time necessary for independent creation. Suppose the adversary can win the game with $f_\mathcal{A}$ (possibly along with $f_\mathcal{A}$'s training dataset $\mathcal{X}_\mathcal{A}$ to achieve $\widehat{s^d} \approx 0$) obtained by running algorithm $\mathcal{A}$ on $vk_\mathcal{V}, f_\mathcal{V}, \mathcal{X}_\mathcal{V}$ that takes time t. Here, we can see that the output of $\mathcal{A}$ is independent of $vk_\mathcal{V}$ because the distributions of verification key are statistically close. In other words, vk's distribution is statistically close to vk' whose marking key $mk' \neq mk$, thus $\mathcal{A}$ cannot distinguish different vk and exploit it for generation. Hence, for arbitrary vk returned by running `KeyGen`, the adversary obtains $f_\mathcal{A}$ (or $\mathcal{X}_\mathcal{A}$) that preserves accuracy on test data (i.e., low generalization error) by running algorithm $\mathcal{A}$ within time t, which opposes the assumption that no such algorithm exists. Therefore, the adversary cannot break the unremovablity.

Unforgeability. We reduce the problem of breaking the requirement to the problem of breaking the commitment. Assume that the adversary can break the unforgeability via algorithm `Forge` that can generate a marking key mk' given vk'. Then, given an arbitrary value c'_V, the adversary can obtain *non-negative* output (i.e., $\texttt{Open}(c'_V, V', r'_V) = 1$, where $(V', r'_V) = \texttt{Forge}(c'_V)$).

Order-preserving Registration. Assume that the adversary has registered the key vk' to the third party, and aims to convince the third party of the ownership, instead of the victim who has registered vk to the third party earlier. The third party can call `Verify` to determine the earlier registration by comparing the registration time. ■

D. Results with Top Confidence Scores

1) Estimation of Dataset Similarity: we assume the victim has access to the top 5 confidence scores. We show the estimation accuracy in Fig. 18. Comparing with the results with full confidence vector, we observe that there is little difference on the estimation accuracy curve, i.e., the constraint of "top 5 confidence score" does not affect our dataset fingerprinting.

2) Estimating Model Similarity: We omit the dataset CIFAR-10 since it only contains 10 classes. For CIFAR-100 and Tiny-ImageNet, we take top 5 and top 10 confidence scores respectively. The results for detecting static modification and classifying finetuned and independent models are shown in Table VII and Table VIII, respectively.