

# Adaptive Domain Inference Attack with Concept Hierarchy

Yuechun Gu, Jiajie He, Keke Chen

{ygu2,jiajieh1,kekechen}@umbc.edu

Trustworthy and Intelligent Computing Lab (TAIC), Computer Science and Electrical Engineering

University of Maryland, Baltimore County

Baltimore, MD, USA

## Abstract

With increasingly deployed deep neural networks in sensitive application domains, such as healthcare and security, it's essential to understand what kind of sensitive information can be inferred from these models. Most known model-targeted attacks assume attackers have learned the application domain or training data distribution to ensure successful attacks. Can removing the domain information from model APIs protect models from these attacks? This paper studies this critical problem. Unfortunately, even with minimal knowledge, i.e., accessing the model as an unnamed function without leaking the meaning of input and output, the proposed adaptive domain inference attack (ADI) can still successfully estimate relevant subsets of training data. We show that the extracted relevant data can significantly improve, for instance, the performance of model-inversion attacks. Specifically, the ADI method utilizes the concept hierarchy extracted from the public and private datasets that the attacker can access and applies a novel algorithm to adaptively tune the likelihood of leaf concepts in the hierarchy showing up in the unseen training data. For comparison, we also designed a straightforward hypothesis-testing-based attack – LDI. The ADI attack not only extracts partial training data at the concept level but also converges fastest and requires the fewest target-model accesses among all candidate methods. Our code is available at <https://anonymous.4open.science/r/KDD-362D>.

## CCS Concepts

• Security and privacy → Privacy protections.

## Keywords

Machine learning, Privacy protection, Membership inference attack, Model inversion attack

## ACM Reference Format:

Yuechun Gu, Jiajie He, Keke Chen. 2025. Adaptive Domain Inference Attack with Concept Hierarchy. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25)*, August 3–7, 2025, Toronto, ON, Canada. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3690624.3709332>

## 1 Introduction

Large-scale deep learning models are increasingly deployed in application domains, playing pivotal roles in sectors where sensitive or proprietary data is used in training the models [3, 32, 33]. These


models might be packaged in API services or embedded into applications, becoming a concerning new attack vector. Recent studies have shown web services are exposed to various types of attacks [12, 25]. One such practical attack is stolen secret credentials or API secret ID. With the stolen credentials, several effective model-targeted privacy attacks, including model inversion (or training data reconstruction) [14, 37], property inference [15, 36], and membership inference attacks [22, 34].

However, we have noticed all these model-targeted attacks depend on a certain level of knowledge about the application domain. (1) Model-inversion (MI) attacks depend on a learning procedure, e.g., a GAN method [38], to progressively adjust seed images from a domain or distribution similar to the training data domain towards most likely training examples. Without this domain knowledge, i.e., with irrelevant auxiliary data, the attack performance can be significantly reduced [17]. (2) Membership-inference attacks (MIA) estimate the possibility of a target sample belonging to the training data of a model. Most MIA attacks<sup>1</sup> assume attackers know the type and distribution of training to train shadow models, which are also important for making sense the MIA result in practice. (3) Property-inference attacks try to uncover global properties of the training data that the model's creator did not intend to reveal. These attacks also need shadow classifiers that are trained on tasks similar to that of the target classifier. Domain knowledge plays an important role in training shadow classifiers.

Thus, one may wonder whether it can protect a model from all these attacks by stripping off the domain-related information in model applications. More specifically, can we achieve the protection goal by packaging the model as an unnamed function call or a service API:  $y = f(x)$ , where an input  $x$  leads to a prediction  $y$ , and no meaning of  $x$  and  $y$  is given? Such a service API mechanism is still convenient and deployable in a cloud-based application ecosystem.

Our recent study shows that even with such a minimal-knowledge setting, attackers can find ways to re-establish knowledge about the application domain. We [17] first attempted to reconstruct the domain information with model accesses only. We used a generative adversarial network (GAN) approach (GDI) to infer which of a set of known candidate datasets is most similar to (or likely from) the target model's domain.

However, the GDI method has several drawbacks. (1) It's dataset-oriented, i.e., assuming datasets similar to the target training data exist in the candidate datasets. Our experiment shows that this method does not work well if the candidate dataset is only partially similar to the target dataset, e.g., only one or a few classes are

 This work is licensed under a Creative Commons Attribution International 4.0 License.

KDD '25, Toronto, ON, Canada

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1245-6/25/08

<https://doi.org/10.1145/3690624.3709332>

<sup>1</sup>Most recently, Carlini et al. [6] propose to use one-side non-member hypothesis testing for approximately identify the membership, which may not need to know the target distribution. However, knowing the domain will help attackers quickly identify true positive samples.

similar. (2) The GAN-based procedure requires excessive accesses to the target model, which may alarm the model service owner.

**Scope of our research.** While GDI targets the dataset-level inference, we wonder whether the record-level inference, i.e., membership inference without knowing the domain, is also possible. We noticed that the offline version of the recently developed Likelihood Ratio Attack (LiRA) [6] meets our requirements, which only needs to train offline shadow models without domain knowledge. We name this first candidate method: offline LiRA domain inference (LDI). However, we found that simply applying LiRA does not work satisfactorily for two reasons. First, it assumes that all out-domain samples perform similarly following a normal distribution regardless of target domain distributions, which may lead to errors for specific target domains. Second, it needs to test every provided record to determine its likely membership, which is expensive. We experimented with improved methods like class-wise sampling, but the result was still unsatisfactory.

By extending the idea of record-level inference to class-level inference, we propose the *adaptive domain inference attack* (ADI) to address the weaknesses of both GDI and LDI. Specifically, the ADI approach can detect likely training examples at the concept (or class) level. ADI makes the extracted domain information more accurate than the GDI’s dataset level. ADI also utilizes the concept hierarchy to direct the inference to stay focused on likely concepts, which avoids inefficient record-level procedures like LDI. We find that ADI requires much fewer model accesses than GDI and LDI and yields better-quality inference, as Section 4.3 shows.

The ADI attack assumes that the attacker has minimal knowledge about the unnamed model API, e.g., a black-box image classification model with only information about the input image size and output probability vector. The attacker tries to estimate the target domain via the existing knowledge about a (possibly relevant) mini-world in the form of concept hierarchy. The process can be described as follows. First, the ADI attack establishes a concept hierarchy from the data pool that the attacker can collect or adopts an existing one, e.g., the ImageNet concept hierarchy (Figure 9 in Appendix C), as long as samples at the leaf concepts are available. Each leaf node of the concept hierarchy represents a concept associated with a cluster of sample images. Each internal node represents a higher-level concept, e.g., a group of relevant concepts. Each node is also associated with the probability of being relevant to the target model’s domain, initially set to equal probability. The ADI algorithm will iteratively adjust the node probabilities with the samples from the randomly sampled leaf concepts and the feedback from the target model prediction. After a few iterations, the probabilities of the relevant leaf concepts are promoted, while irrelevant ones get demoted. Finally, we sample leaf concepts according to their probabilities to generate a candidate dataset to assist model-based attacks.

Our method has several unique advantages. (1) It can identify relevant classes of records among a huge data pool more accurately and efficiently than the dataset-oriented GAN-based domain inference attacks (GDI) [17] and our record-oriented LDI method. The experimental results also show that the hierarchical concept organization is crucial in guiding the algorithm to identify the relevant concept more effectively than flatly organized concepts. In assisting model-inversion attacks, the ADI-extracted datasets work about 20% better than the candidate datasets identified by GDI. (2)

It achieves better attacking results with much fewer model accesses. It works on a small batch of samples per iteration, e.g., 1000 images, and the number of iterations is small, e.g., around 40. Overall, ADI requires orders of magnitude fewer model accesses than GDI and LDI.

The remaining sections include the basic notations and definitions (Section 2), the detailed description of the LDI and ADI attacks (Section 3), the experimental evaluation (Section 4), the related work (Section 5), and our conclusion (Section 6).

## 2 Preliminaries

### 2.1 Definitions

In our context, the machine learning model under attack, denoted as  $f(x)$ , processes input data  $x$  (such as an image) and outputs a confidence vector,  $v = f(x)$ . This vector contains the probabilities that the input belongs to each potential class. The label,  $y$ , corresponds to the class with the highest probability in this vector. The model is trained using a dataset,  $D_T$ , a subset drawn from an unknown “Latent Domain”  $S_T$ . Once the model is trained and deployed, users interact with it, typically via an API. They don’t have any direct access or knowledge about the original training data,  $D_T$ .

**Data pool.** A data pool may consist of multiple training datasets from public or private domains,  $P = \{D_1, D_2, \dots, D_n\}$ , where  $D_i$  contains feature vector and label pairs. The attacker prepares a data pool to perform the attack.

**Dataset similarity.** Our proposed attack extracts data records,  $D_e$ , from the data pool that are closely related to the target training data  $D_T$ . The effectiveness of the attack is measured by the similarity of  $D_e$  to  $D_T$ , i.e.,  $\text{Distance}(D_e, D_T)$ . This similarity is crucial for effective model-based attacks, such as model-inversion attacks [17, 37]. We use the Optimal Transport Dataset Distance (OTDD) [4] to assess this similarity. OTDD measures dissimilarity between datasets using optimal transport distances, providing geometric insight and interpretable correlations.

## 3 Adaptive Domain Inference Attack

The proposed attack aims to directly identify the classes of records in a data pool similar to the training data of the target model. The extracted samples can be used as auxiliary data in model-based attacks. We first discuss the threat model, then briefly describe the record-level LiRA domain inference attack (LDI), and finally present the motivation and detail of ADI.

### 3.1 Threat Modeling

**Involved parties.** The model owner may package the deep learning model as a web service and remove all semantic information from the input and output. The adversary can be any party who is curious about the model and the data used for training the model.

**Adversarial knowledge.** Web service APIs are secured by standard protocols, i.e., via HTTPs (HTTP over TLS or SSL), which ensures communication security. The attacker does not breach the protocol and encryption but uses the stolen credentials to issue regular API calls. The model API accepts only encrypted inputs and outputs, not containing any domain information. We assume attackers have successfully stolen the credentials and have black-box access to the model with limited knowledge about API input/output

information, such as input image size and the number of output classes. Our attack focuses on image classification models, with the API returning a class likelihood confidence vector. However, the attack can be extended to non-image tasks. For label-only outputs, attackers can use strategies like [7] to create pseudo-confidence vectors. Attackers can collect images from diverse sources to build the concept hierarchy, enhancing attack coverage. Larger and more diverse sources increase the likelihood of containing concepts used by the target model, but direct access to the target-domain training and testing datasets is not possible.

**Attack target.** Domain information, such as image types and class definitions, is crucial for model-based attacks. Knowing the domain helps adversaries select suitable auxiliary datasets, enhancing attack efficacy [19, 30, 37]. Our goal is to determine if this information can be derived solely from the exposed model API. Attack performance (relevance of generated data to the original data) and attack cost (number of accesses to the target model) are the main measures for domain inference attacks. The first such attack, GDI [17], is expensive and performs poorly if landmark datasets are only partially relevant to the target domain.

### 3.2 First Candidate – LDI

We wonder whether we can conduct a direct record-level inference without the domain knowledge. Specifically, can we use a membership inference attack (without knowing the domain!) to collect the likely in-domain records and then analyze them to infer the domain information? Most MIA attacks assume the known domain and depend on it to derive shadow models [22], except for a most recent development: the offline version of likelihood ratio attack (LiRA) [6]. Next, we explore the application of offline LiRA in domain inference and summarize its problems.

**Offline LiRA test.** Carlini et al. [6] present both an online and an offline LiRA test. The online LiRA test requires attackers to train multiple shadow models on both in-domain and out-domain datasets, resulting in extremely high computational costs. In contrast, the offline LiRA test does not use in-domain information and relies solely on out-domain samples. More details can be found in the paper [6]. This offline version is well-suited for our domain inference attack, as it does not assume any target domain is known.

Specifically, to test the membership of  $x$ , we measure the probability of observing confidence as high as the target model’s under the null hypothesis that the target point  $(x, f(x))$  is a non-member as follows:

$$\Lambda(x) = 1 - \Pr[Z > \phi(q)], \text{ where } Z \sim \mathcal{N}(\mu_{\text{out}}, \sigma_{\text{out}}^2) \quad (1)$$

where  $q$  represents the difference between the greatest and second greatest confidence scores of the predicted logits. The function  $\phi(q) = \log\left(\frac{1}{1-q}\right)$  applies logit scaling, and  $\mathcal{N}()$  denotes a Gaussian distribution. A one-sided hypothesis testing is conducted to conclude whether the confidence is high enough to reject the null hypothesis, as a member sample’s  $\phi(q)$  value is higher and significantly out of the out-domain samples’  $\phi(q)$  distribution. For simplicity,  $\Lambda > 0.5$  has been used as the threshold to determine the membership of in-domain samples [6].

**Offline LiRA domain Inference.** We can apply the offline LiRA to a record-based domain inference attack as follows. We start with training  $n$  shadow models on  $n$  hypothetical in-domain shadow datasets, each constructed by randomly picking samples from the data pool by flipping a coin. For each shadow dataset, we have the remaining samples in the data pool as out-domain samples, which are used to estimate the parameters  $\mu_{\text{out}}$  and  $\sigma_{\text{out}}^2$ .

Then, we test the membership of each instance in the data pool for the target domain. The instance is used as the input to the target model to generate the corresponding  $\phi(q)$ , and we apply the offline LiRA to determine the membership. Once (likely) member instances are collected, the attacker can analyze them to derive the domain information, e.g., based on the top- $K$  most popular classes or clusters.

We also tested two sampling methods to minimize the model access cost: random sampling and class-balanced sampling. In random sampling, we uniformly randomly select a subset of data records from the data pool. In class-balanced sampling, we take the same number of samples from each class to avoid oversampling large classes and increase the coverage of small classes. It’s specifically implemented by augmenting the less populated classes, i.e., via random rotation and flipping of existing samples in the class. The effect of different sampling strategies is shown in Section 4.4.1. Detailed steps of LDI are provided in Algorithm 1.

---

#### Algorithm 1 LiRA-based domain inference attack

---

**Require:** Model  $f_{D_T}$ , data pool  $D_A$

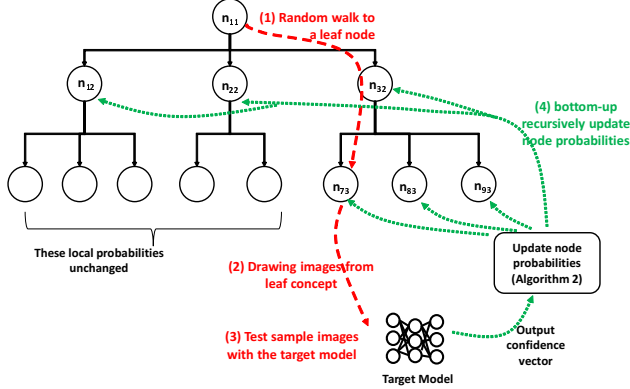
- 1:  $\text{confs}_{\text{out}} \leftarrow \{\}$
- 2:  $\tilde{D}_T \leftarrow \{\}$
- 3: **for**  $i = 1$  to  $N$  **do**
- 4:  $D_{\text{shadow}} \leftarrow D_A$  {Sample a shadow dataset}
- 5:  $f_{\text{out}} \leftarrow \mathcal{T}(D_{\text{shadow}} \setminus \{(x, y)\})$  {Train OUT model}
- 6:  $q \leftarrow f_{\text{out}}(x)_1 - f_{\text{out}}(x)_2$  {Get an observation}
- 7:  $\text{confs}_{\text{out}} \leftarrow \text{confs}_{\text{out}} \cup \{\phi(q)\}$
- 8: **end for**
- 9:  $\mu_{\text{out}} \leftarrow \text{mean}(\text{confs}_{\text{out}})$
- 10:  $\sigma_{\text{out}}^2 \leftarrow \text{var}(\text{confs}_{\text{out}})$
- 11: **for all**  $(x, y) \in D_A$  **do**
- 12:  $q \leftarrow f_{D_T}(x)_1 - f_{D_T}(x)_2$  {Get an observation}
- 13:  $\Lambda \leftarrow 1 - \Pr[Z > \phi(q)], \text{ where } Z \sim \mathcal{N}(\mu_{\text{out}}, \sigma_{\text{out}}^2)$
- 14: **if**  $\Lambda \geq 0.5$  **then**
- 15:  $\tilde{D}_T \leftarrow \tilde{D}_T \cup \{(x, y)\}$
- 16: **end if**
- 17: **end for**

---

Despite using this refined design, we observed that LDI performs unsatisfactorily due to several reasons. First, the offline LiRA method assumes all out-domain instances’ output confidence values  $\phi(q)$  fit in the normal distributions, which may not be precise, as discussed in the original paper [6]. Second, if the target domain instances form only a small portion of the data pool, neither sampling method may ideally extract the likely target domain instances. We show more details in Appendix A.

### 3.3 Concept Hierarchy for ADI

Bearing the problems with LDI in mind, we introduce the adaptive domain inference based on concept hierarchy (ADI). The use of concept hierarchy, depicted in Figure 1, is pivotal to our approach, steering the attack towards probable classes. We will show that a concept hierarchy can help the algorithm better focus on the relevant concepts (and their branches), which is much more effective than a flat concept organization used by LDI. It also serves application scenarios better, as an application dataset likely uses multiple relevant concepts, which can be better captured by the hierarchical structure.



**Figure 1: Concept hierarchy illustration.** The  $i$ -th node at level  $j$  holds probability  $p_{ij}^{(t)}$  at time step  $t$ . Initially, node probabilities are set to  $1/q$ , with  $q$  as the number of child nodes under the node’s parent. In each iteration, Algorithm 2 applies a random walk from the root to a leaf, e.g., a red path. Then, a sample image is drawn from the leaf node and applied to the target model. The result will trigger the probability updates following a green path from the leaf to the root.

We assume a concept hierarchy has been derived from the data pool or borrowed from other applications, while the hierarchy construction method is not the focus of this research. The concept hierarchy is expected to be large enough so that it can intersect or even contain the concepts used by the target model’s potential domain. This hierarchical structure spans  $L$  levels with the root at level 1, and each branch groups similar samples, where naming might not be important. For example, an “skirt” node under “clothes” may lead to subclasses “hoop skirt” and “mini skirt” nodes, while “skirt” under “beef” represents the images of a specific part of beef. Each node carries a *local probability*, starting at  $1/q$  for  $q$  siblings. Traversing from the root to a leaf and multiplying the probabilities on the path, we can get a leaf’s *global probability*. These leaf global probabilities sum to 1, maintaining the algorithm’s integrity. As the attack progresses, the algorithm adjusts local probabilities based on the target model’s feedback, culminating in leaf nodes’ global probabilities that reflect their presence in the target model’s training data.

We have experimented with two simple ways to obtain the concept hierarchy. First, attackers can use an existing image-based

concept hierarchy, e.g., one built from the ImageNet [10] repository, which we have used to show the attack on the DeepFashion model in our experiments. Second, attackers may have an initial guess and collect some datasets that are likely relevant to the target domain. A straightforward approach is to leverage the inherent structure of the collected data pool  $\{D_1, D_2, \dots, D_n\}$  to build a 3-level tree. The nodes at the second level represent individual datasets, and their child nodes correspond to the associated classes or clusters within those datasets. With these concept hierarchies, we have observed ADI performs significantly better than other candidate methods. We are sure that more refined concept hierarchies will further improve our algorithm’s performance.

### 3.4 Details of ADI Attack

Once the attacker has assembled a data pool and formulated a concept hierarchy, they use Algorithm 2 to adjust the node-associated probabilities. The core of the algorithm is an iterative process, as illustrated in Figure 1. Each iteration involves drawing sample images from leaf clusters, using a top-down random walk following the nodes’ local probabilities. The sampling process starts from the root and randomly selects a child branch based on their local probabilities until a leaf is reached. Each sample is then fed into the target model  $f(\cdot)$  that produces a confidence vector,  $v$ .

The crux is how to use the confidence vector to tune the node probabilities. The first step is to determine whether the node is relevant enough to the domain, for which we design two methods.

(1) **LiRA-based method (LiRA-ADI):** We use the offline LiRA test directly to determine the likelihood of membership for the test sample,  $\Lambda$ . If  $\Lambda \geq 0.5$ , we label the sample as a *positive sample*. However, the LiRA test requires a significant setup cost, i.e., training multiple shadow models. Thus, we design the following alternative method.

(2) **Entropy-based method (Entropy-ADI):** The second method is built on the intuition that if an image is similar to a training data class, the confidence probability  $v_i$  for that class,  $y_i$ , significantly exceeds others; if the model fails to recognize the image, class probabilities tend to be similar. We capture this trait by using the concept of *normalized entropy*, and employing a predefined entropy threshold  $\lambda$  to determine if the target model confidently recognizes the sample. Since the range of possible entropy values is determined by the number of classes,  $m$ , i.e.,  $[0, \log_2 m]$ , the normalized entropy function is

$$\tilde{H}(v) = -\frac{1}{\log_2 m} \sum_{i=1}^m v_i \log_2 v_i, \quad (2)$$

where  $v = (v_1, \dots, v_m)$  is the confidence vector. The normalization converts all entropy values, regardless of the number of classes, to the range  $[0, 1]$ , and allows us to establish a generalized algorithm, independent of the target model’s class count. We label samples with entropy  $\leq \lambda$  as *positive samples* since a target-model-recognized sample typically has a low-entropy confidence vector, and those with entropy  $> \lambda$  as *negative samples*. In experiments, we have observed  $\lambda = 0.83$  seems to give the best result.

To unify these two methods, we define the *positive(x, mode)* function with *mode* indicating LiRA-ADI or Entropy-ADI, which tests whether a sample  $x$  is positive. Consequently, we reward the leaf node from which a positive sample was drawn by increasing

the leaf’s local probability. The node probability adjustment is then propagated to sibling and parent nodes, as detailed in the following section. Conversely, a negative example results in a decreased probability for its originating leaf cluster, and the adjustment is also propagated to sibling and parent nodes. Through rounds of these adjustments, we hope that the leaf probabilities are stabilized, reflecting their relevance to the hidden domain of the target model.

---

**Algorithm 2** Overview of ADI ( $f()$ ,  $mode$ ,  $C$ ,  $b$ )

---

**Require:** Target model  $f()$ ,  $mode$ : LiRA-ADI or Entropy-ADI, Batch size  $b$ , Concept hierarchy  $C$  with local probabilities attached to the nodes.

**Ensure:** Updated concept hierarchy  $C$

```

1:  $t \leftarrow 0$ 
2: repeat
3:   Perform random walks on  $C$  from the root to leaves  $b$  times, and draw  $b$  images accordingly. Current batch of images:  $I_t$ 
4:   for  $k = 1$  to  $b$  do
5:      $i \leftarrow$  leaf node index from which  $I_{t,k}$  is drawn
6:     Update  $C$  using  $\text{Adj\_Probs}(C, f(), mode, s, i)$ 
7:   end for
8: until Convergence is achieved for  $I_t$ 
9: return  $C$ 

```

---

The next subsections will give more details for the core steps: the node probability adjustment strategy:  $\text{Adj\_Probs}(C, f(), mode, I_{t,k}, i)$ , and the convergence condition:  $\text{converge}(I_t)$ .

**3.4.1 Concept-probability Adjustment and Propagation.** The algorithm’s central step uses the target model’s output to modify node probabilities within the concept hierarchy. Positive feedback escalates the probability of the originating leaf cluster and its neighboring ones, while negative feedback reduces them.

To clearly describe the probability adjustment and propagation procedure, we give the following notations first. We denote the  $i$ -th node (from left to right) at the  $j$ -th level (1..L from top to down) of the concept hierarchy as node  $n_{ij}$ . As such, each node can be uniquely identified. Let the probability associated with the node at iteration  $t$  be  $p_{ij}^{(t)}$ . The number of siblings of the node can vary – for simplicity, we denote the siblings of the node as a set  $U_{ij}$  and the number of siblings as  $|U_{ij}|$ . We also assume a sampled record,  $s$ , is drawn from the leaf node  $n_{iL}$  at timestep  $t$ . The following update will be recursively applied to the leaf node and its ancestors until the root is reached.

**Target-node probability update.** Let  $\delta(j)$  denote a level-wise adjustment. If the sample is positive, the adjustment is added to the probability of node  $n_{ij}$ ; otherwise, the probability is decreased by  $\delta(j)$ . With the previously defined  $\text{positive}(s, mode)$  function for sample  $s$ , we define:  $w(s, mode) = \text{positive}(s, mode) \cdot 1 : -1$  and

$$p_{ij}^{(t)} = p_{ij}^{(t-1)} + w(s, mode)\delta(j) \quad (3)$$

**Sibling probability update.** Correspondingly, we need to update the siblings’ probabilities to maintain the sum of local probabilities under the parent to be 1. For each positively adjusted node,

its siblings’ probabilities will correspondingly decrease, and vice versa.

$$p_{ij}^{(t)} = p_{ij}^{(t-1)} - w(s, mode)\frac{\delta(j)}{|U_{ij}|}, \quad \text{for } j \in U_{ij} \quad (4)$$

Recursively, the target node is moved up to the parent of node  $n_{ij}$ , and the same target node and sibling probability update procedure is applied until the root is reached.

**3.4.2 Adaptive adjustment  $\delta$  and Probability Rebalancing.** The bottom-up probability adjustment will progressively increase the probabilities of the concepts (and their parents) relevant to the target domain. The updated node probabilities will immediately affect the images sampled in the next iteration in Algorithm 2.

The amount of adaptive adjustment  $\delta$  is designed to attain optimal convergence as detailed in Section 3.4.3. It also controls the intensity of probability update crossing levels. The setting of  $\delta$  is a delicate balance among convergence quality, convergence speed, and the overall coverage of relevant concepts. If it’s too large, the optimal convergence is difficult to reach, and the algorithm may be biased towards a few early visited concepts. If it’s too small, convergence might be slow, increasing the attack’s cost (i.e., the number of model API accesses). The heuristic we use, which has proven successful in experiments, ensures that  $\delta(j)$  decays linearly towards the level  $j$ : the higher the level (the smaller  $j$ ), the less the adjustment is propagated. Moreover, the more nodes the hierarchy has, denoted as  $|C|$ , the less dramatic the adjustment should be to increase the coverage of concepts. We have experimented with  $\delta(j) = j/|C|$  with different scaling factors (Section 4.4) and shown that  $\delta(j) = j/|C|$  is appropriate.

However, merely using the above heuristic is not enough. We observed that aggregation over rounds of adjustment may severely unbalance node probability distributions. To address this, we discount the probability  $p_{ij}^{(t)}$  surpasses the sum of its siblings’ probabilities:  $\sum_{k \in U_{ij}} p_{kj}^{(t)}$ . Specifically, we subtract the mean of the excess value and redistribute it proportionally to the other sibling nodes. This update maintains the sum of probabilities under each node equal to 1 while ensuring different nodes’ probabilities are adjusted proportionally.

The rebalancing strategy is applied as follows if  $p_{ij}^{(t)} > \sum_{k \in U_{ij}} p_{kj}^{(t)}$ . Let  $d = p_{ij}^{(t)} - \sum_{k \in U_{ij}} p_{kj}^{(t)}$ .

$$\begin{cases} p_{ij}^{(t)} = p_{ij}^{(t)} - d \\ p_{kj}^{(t)} = p_{kj}^{(t)} + \frac{d}{|U_{ij}|}, \text{ for } k \in U_{ij} \end{cases}$$

Leveraging the methods as mentioned earlier, we give the adjustment algorithm,  $\text{Adj\_Probs}(C, mode, s, i)$  (Algorithm 3).

**Algorithm 3** Adj\_Probs( $C, f(), mode, s, i$ )

**Require:** Concept hierarchy  $C$ , Target model  $f()$ ,  $mode$ : LiRA-ADI or Entropy-ADI, Image  $s$  is drawn from the leaf node  $n_{iL}$

**Ensure:** Updated concept hierarchy  $C$

```

1: Initialize:  $j \leftarrow L$ 
2: while  $j \neq 1$  do
3:    $p_{ij}^{(t)} \leftarrow p_{ij}^{(t-1)} + w(s, mode) \delta(j)$ 
4:   for all  $k \in U_{ij}$  do
5:      $p_{ij}^{(t)} \leftarrow p_{ij}^{(t-1)} - w(s, mode) \frac{\delta(j)}{|U_{ij}|}$ 
6:   end for
7:   Rebalancing step
8:    $j \leftarrow j - 1$ 
9:    $i \leftarrow$  index of the parent node of the current node
10: end while
11: return Updated concept hierarchy  $C$ 

```

**3.4.3 Convergence Condition.** Once the relevant concepts get boosted probabilities, the late iterations will more likely fetch the samples relevant to target domains. This positive feedback continues until convergence. A critical question is when we should stop the iterations. With the rebalancing strategy, the relevant node’s probability eventually converges to  $p_{ij}^{(t)} \approx \sum_{k \in U_{ij}} p_{kj}^{(t)}$ . However, it’s not sufficient to serve as a global convergence condition.

We design a global strategy as follows. The intuition is that if the majority of the sampled batch  $I_t$  fits the target model  $f()$  well, i.e., for most of them the normalized entropy  $\tilde{H}(f(I_{tk})) \leq \lambda$  in Entropy-ADI, or the probability  $\Lambda(f(I_{tk})) \leq 0.5$  in LiRA-ADI, we consider the concept probability adjustment converges. Thus, the converge function is a Boolean function defined as follows.

$$converge_{entropy}(I_t) = \frac{1}{b} \sum_k \tilde{H}(I_{tk}) \leq \lambda? 1 : 0 \quad (5)$$

OR

$$converge_{LiRA}(I_t) = \frac{1}{b} \sum_k \Lambda(I_{tk}) \geq 0.5? 1 : 0 \quad (6)$$

In the entropy-based method, we have carefully evaluated the setting of  $\lambda$  in experiments and found that  $\lambda = 0.83$  works best with different experimental datasets.

It’s important to note that the convergence will fail when the concept hierarchy is not large enough to cover any class of the target domain. As observed in experiments, most convergences happen around 50 iterations. We may use a relaxed upper bound, e.g., 100 iterations, to determine whether convergence is not possible, i.e., the concept hierarchy may not contain any class of the target domain.

## 4 Experiments

ADI aims to address the limitations of the GAN-based GDI method [17]: it’s difficult to identify candidate datasets at the dataset level similar to the target training data, and excessive target model accesses are also disadvantageous in private model application scenarios, e.g., the attack is more likely to be detected. Our experiments demonstrate that ADI can effectively mitigate these limitations. Specifically, the experiments will achieve the following goals. (1)

We demonstrate that our attacks provide more accurate domain estimations compared to GDI on high-resolution, ImageNet-related datasets and complex concept hierarchies. (2) Our attacks require significantly fewer accesses to the target model to achieve superior results, which is crucial for attacks in privacy-sensitive environments. (3) We have thoroughly evaluated the impact of various parameter settings on ADI’s performance.

### 4.1 Setup

**Concept hierarchies and datasets.** We tested three methods for creating concept hierarchies. (1) We used a dataset-based hierarchy and synthesized domain datasets for parameter tuning. The three-layer hierarchy includes MNIST [11], EMNIST [8], LFW [23], CIFAR10, CIFAR100 [27], CINIC-10 [9], FASHION-MNIST [35], and CLOTHING [16]. Each dataset is an intermediate node, with classes as leaf concepts. Images are scaled to 36x36x3. For mixed-domain training, we randomly selected 10, 20, and 30 classes from the 8 datasets to build synthesized training sets. Unselected data formed the adversaries’ data pool and hierarchy.

(2) We also experimented with a larger hierarchy from ImageNet-1k and three ImageNet-related datasets: ImageNETTE [21], ImageWoof [21], and DeepFashion [29]. The hierarchy includes 1000 leaf nodes (ImageNet-1k classes) and up to 10 intermediate layers<sup>2</sup>. DeepFashion uses 10 classes from the "Clothing" branch of ImageNet. ImageNETTE includes 10 easily classified ImageNet classes, and ImageWoof includes dog breeds from ImageNet. These examples demonstrate the functionality of a larger hierarchy. For more details, see Appendix C.

**Target models.** We used ResNet-18 [20] for the three synthesized datasets, training with an 8:2 split, repeated 10 times to observe variance. The training used SGD with a learning rate of  $10^{-2}$ , batch size of 100, momentum of 0.9, and learning rate decay of  $10^{-4}$ .

For the three high-resolution ImageNet-related datasets, we used ResNet-152, with the same split, repetitions, and hyperparameters as the ResNet-18 models. All models were trained using the Fast Forward Computer Vision (FFCV) pipeline for efficiency [28].

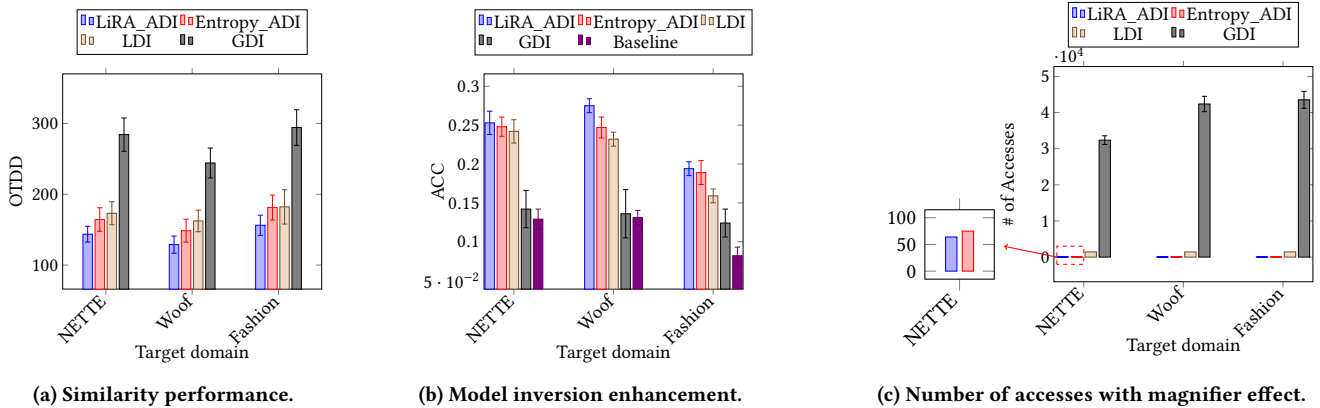
**Shadow models.** Shadow models for depicting out-domain samples’ output confidence vectors are crucial for the hypothesis-based membership inference attack and, thus, vital to both LDI and ADI. Since class numbers vary, we separate the data pool by target domain classes and train shadow models accordingly. For a  $k$ -class target domain, we divide the attack pool into multiple  $k$ -class sub-domains and randomly sample  $n$  sub-domains to train  $n$  shadow models per sub-domain. We set  $n = 128$  for synthesized datasets and  $n = 256$  for ImageNet-related datasets, as suggested in [6]. For details on shadow models’ impacts, see Appendix B.

### 4.2 Evaluation Metrics

**Dataset similarity.** As mentioned in Section 2, we employ the Optimal Transport Dataset Distance (OTDD) to quantify dataset similarity between the original target dataset and the dataset extracted by the attack. Lower OTDD values indicate greater similarity between the estimated dataset and the original domain.

<sup>2</sup>The ImageNet-1K hierarchy is available at our code link.





**Figure 2: Comparison of performance on high-resolution datasets and large-scale concept hierarchy. ADI outperforms other domain inference attacks in both efficiency and efficacy.**

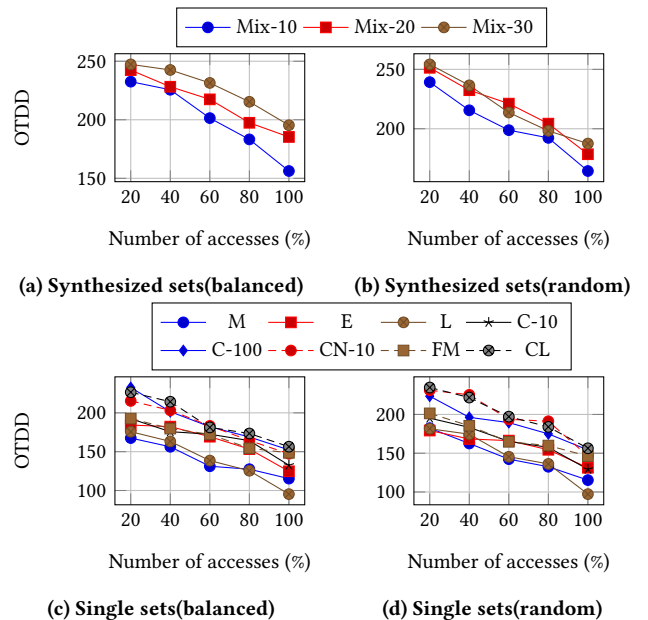
**Performance of model inversion attack.** The GDI, LDI, and ADI estimated data will be used as the auxiliary datasets to the model-inversion attack – if the estimated data is similar to the original domain, the accuracy of the model-inversion attack will be significantly boosted. We use the original model-inversion attack designed by Fredrikson et al. [14], the effectiveness of which is evaluated by the quality of its reconstructed training dataset. We use the target model to recognize (i.e., classify) the reconstructed images – the higher the classification accuracy, the better the reconstructed data and thus the better the estimated data.

### 4.3 Results on ImageNet-Related Datasets

We show the methods’ attacking performance on the three ImageNet-related datasets. Due to page limitations, we include the results on synthesized datasets in Appendix D. We use the parameters that generate the best-estimated datasets with the smallest OTDD scores for the attacking methods. For LDI, we select 100% of instances from balanced classes. For ADI, we use the optimal parameter settings for  $\lambda$  and  $\delta(j)$ . We will discuss the parameter settings in later sections.

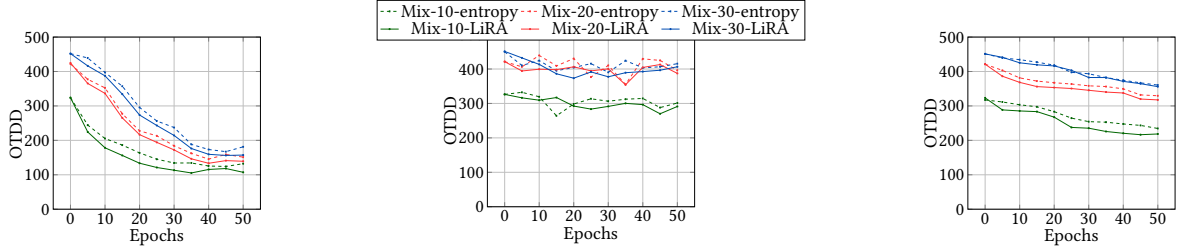
**Quality of estimated domain.** To observe GDI’s performance, we split ImageNet-1K into 100 datasets with 10 similar classes each to build the landmark datasets, such as 10 dog breeds, 10 types of clothing, etc. The results in Figure 2a show that LiRA-ADI performs the best with the smallest OTDD score, and Entropy-ADI also outperforms LDI and GDI.

**Enhancing model inversion attacks.** Model inversion attacks require knowledge of the target domain to implement or enhance the attack. We are also interested in how the estimated domain can enhance model inversion attacks. We recover 1000 images for each class of the target model using model inversion attacks and test the target model’s performance on these recovered datasets. In the baseline scenario, we initialize the generated images randomly. In other experiments, we randomly pick images from the estimated domain as the initial images. Figure 2b shows the enhancement of model inversion attacks. Both ADI methods outperform other attacks with LDI ranked next.



**Figure 3: OTDD constantly decreases with the increasing size of tested samples in LDI. Dataset Names: M-MNIST, E-EMNIST, L-LFW, C-CIFAR, CN-CINIC, FM-FashionMNIST, CL-CLOTHING, NETTE-ImageNETTE, Woof-ImageWoof, Fashion-DeepFashion.**

**Model Accesses.** One motivation for designing our attacks is to reduce the substantial number of accesses required by GDI. Figure 2c illustrates the number of accesses needed for each attack to reach convergence. We set the batch size for drawing instances and interacting with the target model is 1000. As shown in the results, ADI requires significantly fewer accesses to implement compared to GDI and LDI. The large number of accesses needed by GDI is due to the necessity of training a generative model for each dataset, which involves multiple interactions with the target model. LDI’s model access is critically related to the result quality. We show in



(a) Setting  $\delta = \frac{j}{|C|}$ : Proper  $\delta$  makes quick convergence. (b) Setting  $\delta = \frac{10j}{|C|}$ : Large  $\delta(j)$  causes oscillation, difficult to converge. (c) Setting  $\delta = \frac{j}{10|C|}$ : Lower  $\delta$  slows convergence.

Figure 4: Examining the effects of  $\delta$  configurations.

the next section that LDI needs to test almost all samples in the data pool to get the best-performing results, which can be expensive.

#### 4.4 Parameter Settings

In this section, we include more details for determining the optimal parameter settings for LDI and ADI methods.

**4.4.1 Effect of Sampling Methods for LDI.** To understand the effect of different sampling methods on the LDI results, we progressively increase the number of tested samples. For simplicity, we augmented small classes to make the class sizes even so that we can use percentages to represent the sampling progress for both uniform sampling and class-balanced sampling. The results are shown in Figure 3. We conducted experiments on two types of target domains (and target models): a synthesized target domain consisting of randomly selected classes in the dataset pool, denoted by Mix-10, Mix-20, and Mix-30; uniform samples from each dataset, denoted by the corresponding dataset. The two sampling methods are used to generate the test samples, aiming to reduce the cost of LDI, denoted by “random” and “balanced”. However, we found that the LDI’s performance is almost linearly related to the number of test samples. It does not reach the peak performance until all samples are tested. This pattern keeps across all datasets and sampling methods, which raises the concern that to achieve the best LDI performance, the number of model accesses will be significantly high. However, even with such a high cost, LDI still performs 20% worse than ADI.

**4.4.2 ADI Parameter Settings.** The ADI algorithm contains several parameters to be experimentally explored and determined, including the threshold of the normalized entropy,  $\lambda$  in the entropy-based method, and the layer-wise adaptive probability adjustment  $\delta$  in both entropy and LiRA-based method. We also want to observe the effect of probability rebalancing and assess the benefits of using concept hierarchies against a flat concept list.

**$\delta$  settings.** The layer-wise probability adjustment  $\delta$  plays a crucial role in the speed, quality of convergence, and attack efficiency. We found that  $\delta(j) = j/|C|$  for nodes at Level  $j$  and the total number of concepts  $|C|$  works reasonably well according to the heuristic mentioned in Section 3.4.1(Figure 4a). We have also tested variants of  $\delta$  settings, i.e.,  $\delta(j) = \frac{10j}{|C|}$  and  $\delta = \frac{j}{10|C|}$ . With  $\delta(j) = \frac{10j}{|C|}$ , we observed significant oscillations at higher OTDD levels, indicating the attack does not converge well (Figure 4b). Conversely, reducing

$\delta(j)$  to  $\frac{j}{10|C|}$  resulted in a substantial slowdown in convergence, increasing the attack cost (Figure 4c).

**Effect of rebalancing.** Without probability rebalancing, ADI may lead to a biased concept distribution, i.e., focusing on increasing the probabilities of a few branches of the concept hierarchy due to their already high probabilities. Other methods, such as fine-tuning the  $\delta$  values, may help address this issue. However, we find the rebalancing method works extremely well. Experiments show a remarkable improvement in ADI performance with rebalancing, as shown in Figure 5. Without rebalancing, the ADI gives unstable results. Due to the biased adjustment towards certain branches, the results are often stuck at suboptimal levels.

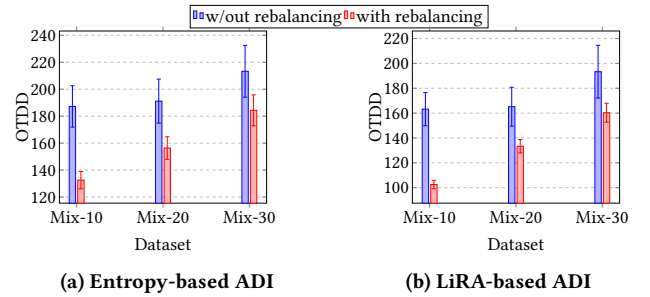
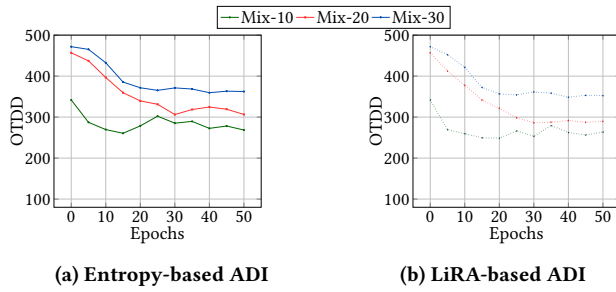


Figure 5: Comparison of entropy-based and LiRA-based methods with and without rebalancing on different datasets.

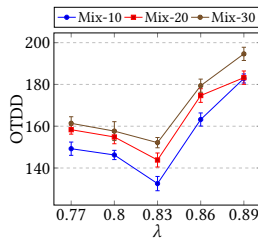
**Effect of Concept Hierarchy.** We wonder how the multi-layer hierarchical structure may affect the ADI algorithm, compared to a flat list of concepts. We construct the flat structure by removing all the internal nodes of the concept tree, i.e., the root node pointing to all leaves directly, so that the ADI algorithm still works without modification. The algorithm still converges, but with much worse quality as shown in Figure 6. Without the hierarchically organized concepts, the ADI estimated domains yield larger OTDD values, meaning less similar to the desired domain distributions, and the variances are also larger. The result indicates that the intermediate nodes can direct the probability adjustments more to the focal concepts, resulting in a better-estimated concept distribution. In contrast, the flat concept structure disperses the probability adjustments.





**Figure 6: OTDD values over epochs for ADI with flattened concept hierarchy.**

$\lambda$  **settings in entropy-based ADI.** The  $\lambda$  setting serves as the threshold, indicating (1) the target model predicts the extracted sample with high confidence, i.e., a positive sample, and (2) the overall quality of the batch of extracted samples, i.e., the convergence condition. Its setting is critical for the entropy-based ADI attack to converge quickly toward a high-quality result. We have conducted a set of experiments to investigate the optimal setting. Figure 7 shows variable settings of  $\lambda$  and  $\lambda = 0.83$  gives the best-extracted datasets among others. It’s worthy noting that LiRA-based ADI does not need this parameter, which makes it more stable in practice.



**Figure 7: The setting of  $\lambda$  significantly impacts the quality of the attack.  $\lambda = 0.83$  yields the best outcomes.**

## 5 Related Work

Machine learning models in fields like intrusion detection and healthcare are increasingly exposed to cyber threats through API services [1, 2, 5, 18, 31–33]. Significant threats include model inversion and inference attacks, which compromise training data privacy. Model inversion attacks aim to recreate training examples using auxiliary data, with recent GAN-based methods achieving high-quality results [37]. Membership inference attacks identify whether specific data was used in training [7, 24, 34], while property inference attacks reveal population-level attributes [15, 36]. Both strategies rely on understanding the target model’s training data distribution. The Fidel attack on federated learning exploits neuron data to infer previous activations, posing a security risk that depends on having an auxiliary dataset resembling the victim’s training data [13].

While the assumption of an attacker knowing the auxiliary data or domain distribution may not hold for a breached private model API service, it is critical to investigate if accessing an unnamed

model API only can also reveal domain information. A recent GAN-based model-domain inference (GDI) attack aims to address this [17]. However, GDI does not work satisfactorily when only a few parts of candidate datasets are relevant to the target dataset, and it requires excessive model accesses. Our proposed ADI attack directly addresses these two problems with the GDI attack.

## 6 Conclusion

Deep neural network models can be exploited by model-targeted attacks, e.g., model inversion and membership inference attacks. However, attackers cannot apply these attacks effectively without knowing the model’s domain information. A possible protection approach is to package a sensitive model as an unnamed function call/API service, hiding the model’s domain knowledge from attackers. However, we show that the Adaptive Domain Inference (ADI) attack can still identify the classes of samples relevant to the target model’s domain by only accessing the unnamed model. By utilizing the concept hierarchy extracted from a large data pool, ADI’s iterative procedure can efficiently and effectively tune the attack towards the concepts likely included by the target model’s training data. ADI surpasses the compared domain inference attacks GDI and LDI by a large margin in the extracted datasets’ quality and significantly fewer model accesses.

**Acknowledgement.** This material is based upon work supported by the National Science Foundation under Grant No. (2232824).

## References

- [1] Mahmood A Al-Shareeda, Mohammed Anbar, Selvakumar Manickam, and Izzan H Hasbullah. 2020. Review of prevention schemes for man-in-the-middle (MITM) attack in vehicular ad hoc networks. *International Journal of Engineering and Management Research* 10 (2020).
- [2] Zainab Alkhalil, Chaminda Hewage, Liqaa Nawaf, and Imtiaz Khan. 2021. Phishing attacks: A recent comprehensive study and a new anatomy. *Frontiers in Computer Science* 3 (2021), 563060.
- [3] Joshua Alspector and Thomas Dietterich. 2020. DARPA’s Role in Machine Learning. *AI Magazine* 41, 2 (Jun. 2020), 36–48. doi:10.1609/aimag.v41i2.5298
- [4] David Alvarez-Melis and Nicolo Fusi. 2020. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems* 33 (2020), 21428–21439.
- [5] Bharat Bhushan, Ganapati Sahoo, and Amit Kumar Rai. 2017. Man-in-the-middle attack in wireless and computer networking—A review. In *2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA)(Fall)*. IEEE, 1–6.
- [6] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. 2022. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1897–1914.
- [7] Christopher A Choquette-Choo, Florian Tramèr, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International conference on machine learning*. PMLR, 1964–1974.
- [8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2921–2926.
- [9] Luke Nicholas Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. 2018. CINIC-10 is not ImageNet or CIFAR-10. *CoRR* abs/1810.03505 (2018). arXiv:1810.03505 <http://arxiv.org/abs/1810.03505>
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- [11] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.
- [12] Josue Diaz-Rojas, Jorge Ocharán-Hernández, Juan Carlos Pérez-Arriaga, and Xavier Limón. 2021. Web API Security Vulnerabilities and Mitigation Mechanisms: A Systematic Mapping Study. 207–218. doi:10.1109/CONISOFT52520.2021.00036
- [13] David Enthoven and Zaid Al-Ars. 2022. Fidel: Reconstructing private training samples from weight updates in federated learning. In *2022 9th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 1–8.

- [14] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.
- [15] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 619–633.
- [16] Alexey Grigorev. 2020. Clothing dataset (full, high resolution). <https://www.kaggle.com/datasets/agrigorev/clothing-dataset-full>. Accessed: 2020-10-21.
- [17] Yuechun Gu and Keke Chen. 2023. GAN-based domain inference attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 14214–14222.
- [18] Brij B Gupta, Aakanksha Tewari, Ankit Kumar Jain, and Dharna P Agrawal. 2017. Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications* 28 (2017), 3629–3654.
- [19] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. [n. d.]. LOGAN: Membership Inference Attacks Against Generative Models. *Proceedings on Privacy Enhancing Technologies* 2019, 1 ([n. d.]), 133–152.
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [21] Jeremy Howard. [n. d.]. ImageNETTE. <https://github.com/fastai/imagenette/>
- [22] Hongsheng Hu, Zoran Salic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.
- [23] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. 2008. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on faces in Real-Life Images: detection, alignment, and recognition*.
- [24] Bo Hui, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. 2021. Practical blind membership inference attack via differential comparisons. *arXiv preprint arXiv:2101.01341* (2021).
- [25] Mohd Ibrahim, Zolidah Kasiran, and Muhammad Azizi Mohd Ariffin. 2020. API Vulnerabilities In Cloud Computing Platform: Attack And Detection. *International Journal of Engineering Trends and Technology* 1 (10 2020), 8–14. doi:10.14445/22315381/CATIIP202
- [26] Yigitcan Kaya and Tudor Dumitras. 2021. When does data augmentation help with membership inference attacks?. In *International conference on machine learning*. PMLR, 5345–5355.
- [27] A Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront* (2009).
- [28] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Mądry. 2023. FFCV: Accelerating training by removing data bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12011–12020.
- [29] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. 2016. Deep-fashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1096–1104.
- [30] Tomoya Matsumoto, Takayuki Miura, and Naoto Yanai. 2023. Membership inference attacks against diffusion models. In *2023 IEEE Security and Privacy Workshops (SPW)*. IEEE, 77–83.
- [31] Bhargav Pingle, Aakif Mairaj, and Ahmad Y Javaid. 2018. Real-world man-in-the-middle (MITM) attack implementation using open source tools for instructional use. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, 0192–0197.
- [32] Santiago Romero-Brufau, Daniel Whitford, Matthew G Johnson, Joel Hickman, Bruce W Morlan, Terry Therneau, James Naessens, and Jeanne M Huddleston. 2021. Using machine learning to improve the accuracy of patient deterioration predictions: Mayo Clinic Early Warning Score (MC-EWS). *Journal of the American Medical Informatics Association* 28, 6 (2021), 1207–1215.
- [33] Khader Shameer, Kipp W Johnson, Alexandre Yahi, Riccardo Miotto, Li Li, Doran Ricks, Jebakumar Jebakaran, Patricia Kovatch, Partho P Sengupta, Sengupta Gelijns, et al. 2017. Predictive modeling of hospital readmission rates using electronic medical record-wide machine learning: a case-study using Mount Sinai heart failure cohort. In *Pacific Symposium on Biocomputing 2017*. World Scientific, 276–287.
- [34] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.
- [35] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. doi:10.48550/ARXIV.1708.07747
- [36] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. 2021. Leakage of dataset properties in {Multi-Party} machine learning. In *30th USENIX security symposium (USENIX Security 21)*. 2687–2704.
- [37] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. 2020. The secret revealer: Generative model-inversion attacks against deep

neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 253–261.

- [38] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. 2020. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 253–261.

## Appendix A: Bias in LDI

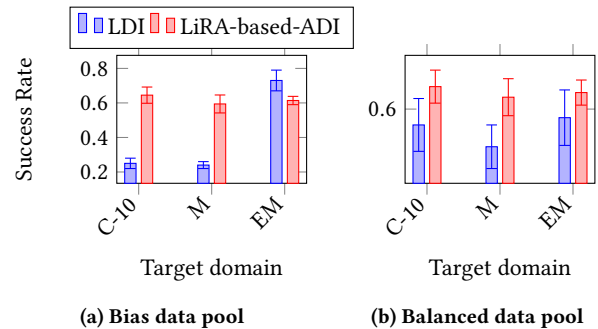
LDI's false negative rate (FNR) lowers its success when the non-target set is much larger than the target. Specifically, the estimated domain is:

$$n_{\text{estimated}} = \text{TPR} \cdot n_{\text{target}} + \text{FNR} \cdot n_{\text{non-target}},$$

and the attack success rate is:

$$\text{Success Rate} = \frac{\text{TPR} \cdot n_{\text{target}}}{\text{TPR} \cdot n_{\text{target}} + \text{FNR} \cdot n_{\text{non-target}}}.$$

When  $n_{\text{non-target}} \gg n_{\text{target}}$ , even a modest FNR sharply reduces the success rate. An experiment on half of CIFAR-10, MNIST, and EMNIST (the other half plus extra datasets in the pool) confirms this effect, especially given EMNIST's large size. A reduced EMNIST or LiRA-based ADI partly alleviates the problem.



**Figure 8: Success rate of target domains shows that the bias of the data pool significantly impacts the dataset.**

Experimental results support our theoretical analysis. As shown in Figure 8a, if we do not reduce the size of EMNIST, the success rates of LDI for CIFAR-10 and MNIST are significantly decreased, while the success rate for EMNIST is much higher. In contrast, ADI demonstrates resistance to the bias in the data pool. Figure 8b presents the results of both attacks with the size of EMNIST reduced to be similar to that of CIFAR-10 and MNIST.

## Appendix B: Impact of Shadow Models.

In Section 4.1, we introduced membership inference attacks that depend on training multiple shadow models for the target domain. Under domain inference, adversaries do not know the target domain, so we train shadow models on the entire data pool. One might wonder if restricting training to sub-domains (e.g., only CIFAR or only MNIST) would still be effective. As shown in [26], membership inference can remain viable even if shadow models are not trained on the exact target domain. Thus, we hypothesize that ADI and LDI should also provide meaningful results in such scenarios.

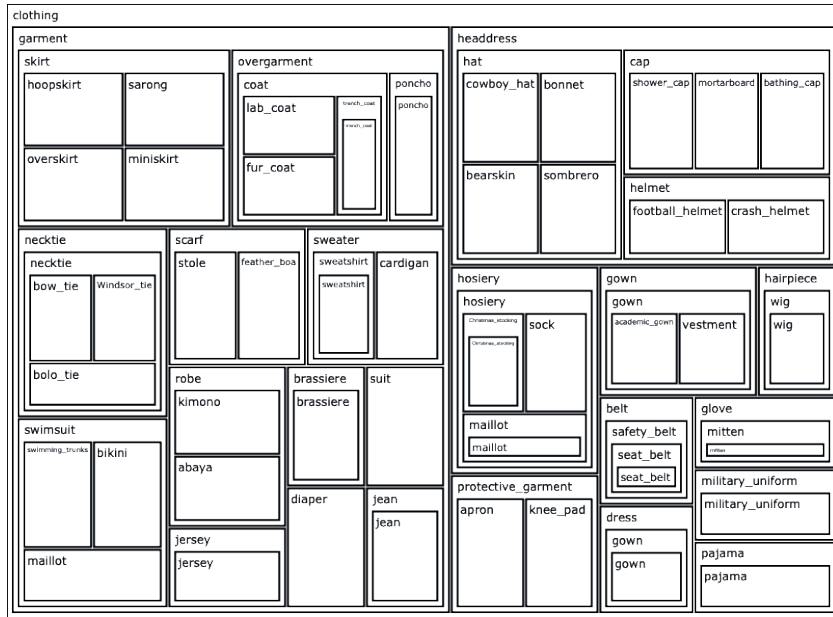


Figure 9: Lower level concept hierarchy of ImageNet-1K.

Concept	Times
work_ware-clothing-covering-artifact-whole	1
skirt-clothing-covering-artifact-whole	1
swimsuit-clothing-covering-artifact-whole	1
garment-clothing-covering-artifact-whole	7

(a) With concept hierarchy.

Concept	Times
fur_coat	2
jeans	3
life_boat	2
Restaurant	1
Mask	2

(b) Without concept hierarchy.

Table 1: The quality of identified concepts.

To test this, we trained shadow models solely on CIFAR-10, MNIST, and Fashion-MNIST, while our target domain was a “Mix-10” set of random classes from the synthesized datasets. Figure 10 demonstrates that training on the full data pool is more effective. Limiting training to one sub-domain lowers attack performance and increases its uncertainty due to the randomness of target domains. For instance, if Mix-10 consists mainly of CIFAR-like classes, shadow models trained on MNIST-like domains perform poorly.

### Appendix C: Straightforward Results on ImageNet Concept Hierarchy

We present the straightforward results of LiRA-based ADI attacks on the DeepFashion-trained model. The attack was repeated 10 times, identifying the branch with the highest probability as the

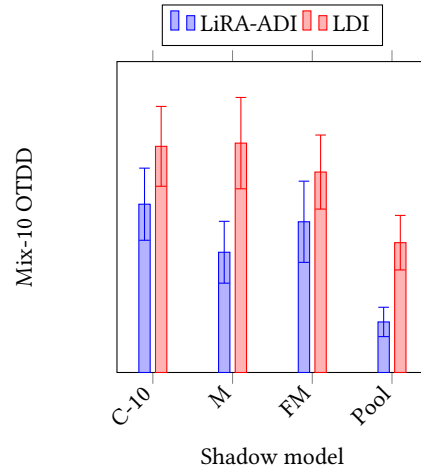


Figure 10: Influence of shadow models on Mix-10 OTDD. “Pool” indicates the shadow model is trained on the entire data pool. Shadow models trained on specific sub-domains decrease the performance of both methods and show higher variability (larger error bars) compared to training shadow models on the entire dataset.

model’s estimated domain. Results with the concept hierarchy (Table 1a) show the attacker accurately deduces the domain as clothing-related items. Without the hierarchy (Table 1b), the estimation is less precise, with non-clothing items like ‘restaurant’ and ‘mask’ causing ambiguity. To further clarify, Figures 9 visualize the concept hierarchies used in our experiments.

## **Appendix D: Results on Synthesized Datasets**

The results in Figures 11, 12, and 13 show that LiRA-based ADI performs the best, followed by entropy-based ADI and LDI. GDI produces meaningful results only on single datasets.

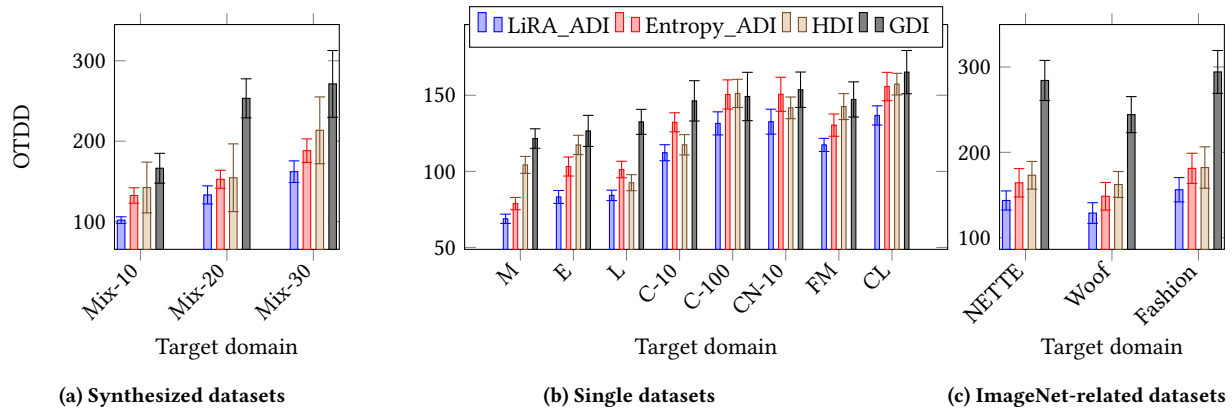


Figure 11: OTDD between the extracted dataset and the target domain.

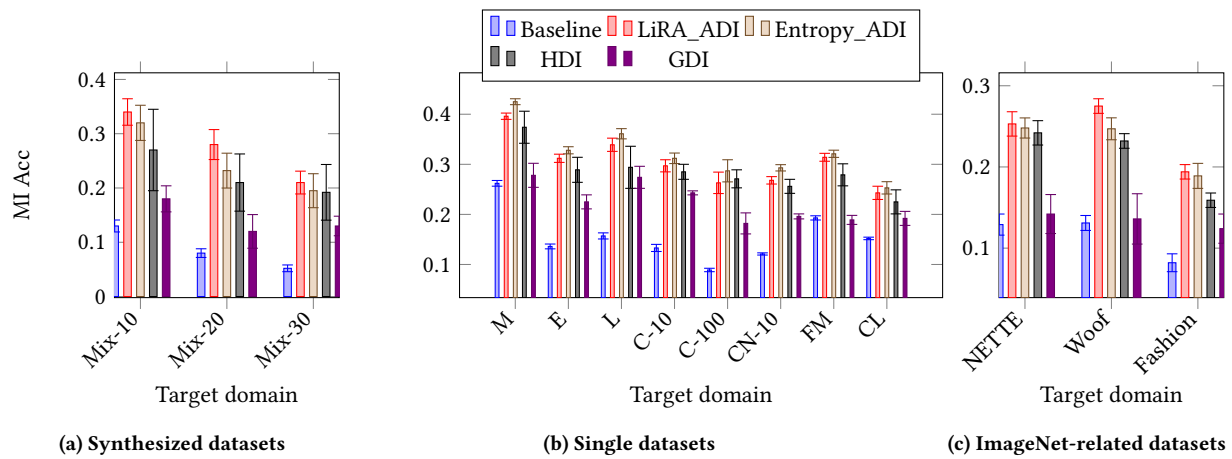


Figure 12: Enhancement to model inversion attacks by estimated domain. ADI and HDI perform well in three scenarios. GDI only performs good in the second scenario.

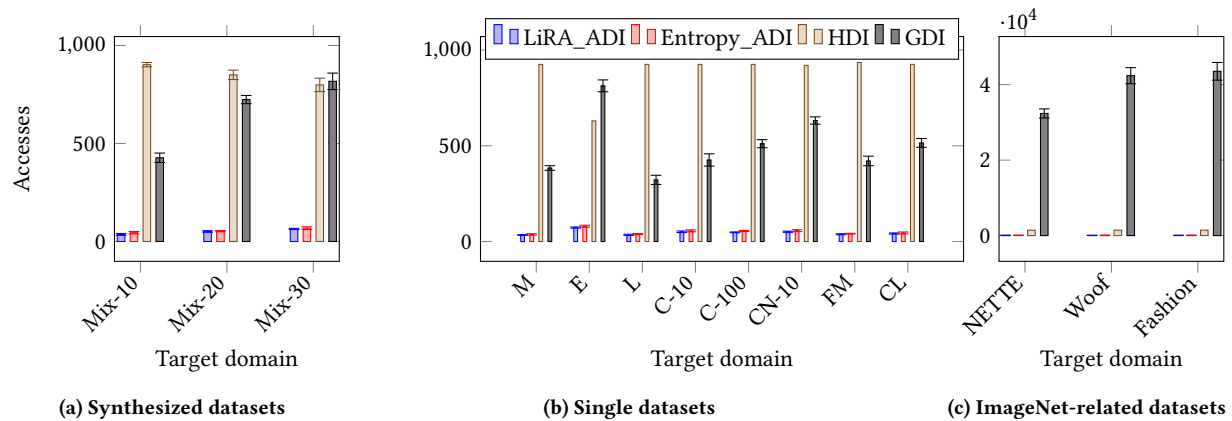


Figure 13: Number of access. ADI needs the smallest amount of access. Both HDI and GDI suffer from huge amounts of access.