Data Isotopes for Data Provenance in DNNs

Emily Wenger
ewenger@uchicago.edu
The University of Chicago

Xiuyu Li
xiuyu@berkeley.edu
UC Berkeley

Ben Y. Zhao
ravenben@uchicago.edu
The University of Chicago

Vitaly Shmatikov
shmat@cs.cornell.edu
Cornell Tech

Abstract

Today, creators of data-hungry deep neural networks (DNNs) scour the Internet for training fodder, leaving users with little control over or knowledge of when their data, and in particular their images, are used to train models. To empower users to counteract unwanted use of their images, we design, implement and evaluate a *practical* system that enables users to detect if their data was used to train a DNN model for image classification. We show how users can create special images we call *isotopes*, which introduce "spurious features" into DNNs during training. With only query access to a model and no knowledge of the model-training process, nor control of the data labels, a user can apply statistical hypothesis testing to detect if the model learned these spurious features by training on the user's images.

Isotopes can be viewed as an application of a particular type of data poisoning. In contrast to backdoors and other poisoning attacks, our purpose is not to cause misclassification but rather to create tell-tale changes in confidence scores output by the model that reveal the presence of isotopes in the training data. Isotopes thus turn DNNs' vulnerability to memorization and spurious correlations into a tool for data provenance. Our results confirm efficacy in multiple image classification settings, detecting and distinguishing between hundreds of isotopes with high accuracy. We further show that our system works on public ML-as-a-service platforms and larger models such as ImageNet, can use physical objects in images instead of digital marks, and remains robust against several adaptive countermeasures.

Keywords

datasets, neural networks, provenance tracking

1 Introduction

As machine learning (ML) systems grow in scale, so do the datasets they are trained on. State-of-the-art deep neural networks (DNNs) for image classification are trained on hundreds of millions or billions of inputs [6, 65, 81]. Often, training datasets include users' public and private images, collected with or without users' consent. Examples include image analysis models trained on photos from Flickr [65] and companies like Clearview.ai training facial recognition models on photos scraped from social media [26].

Today, users have no agency in this process, beyond blindly agreeing to the legal terms of service for social networks, photo-sharing websites, and other online services. Even when users give permission for use of their images, they have little control over how those images may later be shared or disseminated [37]. Beyond searches through specific public datasets like LAION-5B [33], non-expert users have no systematic way to check whether their data was used to train a model [65].

In this paper, we design, implement, and evaluate a practical method that enables users to detect if their images were used to train an image classification DNN model, with only query access to the model and no knowledge of its labels or parameters. Our main idea is to have users introduce special inputs we call *isotopes* into their own data. Like their chemical counterparts, isotopes are similar to normal user data, with a few key differences. Our isotopes are crafted to contain "spurious features" that the model will (mistakenly) consider predictive for a particular class during training. Isotopes are thus amenable to a new type of inference: a user who knows the isotope features can tell, by interacting with a trained model, whether images marked with these features were part of its training dataset or not. Similar inference attacks, such as membership inference [63], are typically interpreted as attacks on the privacy of training data. Helped by the propensity of DNN models to learn spurious correlations, we turn them into an effective tool for tracing data provenance.

Our contributions. We present a practical data isotope scheme that can be used to trace image use in real-world scenarios (e.g., tracing if photos uploaded to a social website are used for DNN training). The key challenge is that users neither know, nor control the supervised classification tasks for which their images may be used as training fodder. While users are free to modify the content of their images, they do not select the corresponding classification labels, nor know the other labels, nor have any visibility into the models being trained. This precludes the use of "radioactive data" [56], "backdoor" techniques [29], and other proposed methods for dataset watermarking (see discussion in §2.3).

Our method creates isotopes by blending out-of-distribution features we call *marks* into images. When trained on these isotopes, a model learns to associate one of its labels with the spurious features represented by the mark. By querying the model's API, a user can verify that the presence of the mark in a test image causes a statistically significant increase in the probability of a low-likelihood output label. Formally, our verification procedure uses statistical hypothesis testing to determine if the model assigns a consistently higher probability to a certain class when the mark is present, independently of other image features. Success implies that the user's marked isotopes must have been present in the model's training dataset. Our method is designed so it can be used by non-ML experts. It does not require users to train shadow or surrogate models, nor compute or analyze gradients of publicly available models. Our key contributions are as follows:

- We propose **a novel method for data provenance in DNN models** using "isotope" data to create spurious correlations in trained models (§3, §4), and a technique for users to detect if a model was trained on their isotope data.
- We **demonstrate the efficacy of our isotope scheme on several benchmark tasks**, including the facial recognition tasks `PubFig` and `FaceScrub`, and show that it remains effective even when multiple users independently add isotopes to their respective data (§5). Despite the potential challenge of having a model learn many isotope-induced spurious features, we find that our verifier can detect and distinguish up to 215 `FaceScrub` isotopes with high accuracy and few false positives, with minimal impact on normal model accuracy.
- We show that **physical objects can act as isotope marks with up to 95% accuracy** (§6), demonstrating that our scheme works even if users cannot digitally modify their images (e.g., when images from surveillance cameras are used to train facial recognition models).
- We **evaluate isotope performance in realistic settings** (§7), including larger models such as `ImageNet` and ML-as-a-service platforms such as Google's Vertex AI. Isotopes have 97% detection accuracy in ImageNet and 89% in Vertex.
- Finally, we **evaluate several adaptive countermeasures** that an adversarial model trainer may deploy against isotopes (§8). All of them either fail to disrupt isotope detection, or incur very high costs in false positives or reduced model accuracy, or both.

Limitations. While effective, our approach has limitations. First, it adds visible modifications to a subset of the user's images, and only certain types of isotope marks are effective. Future work should aim to make marks subtler and investigate a broader set of features that can be used as marks. Second, this paper focuses on images; application of our techniques to other domains (e.g., text generation) is important future work. Finally, while we evaluate several natural countermeasures (§8), future adaptations by model trainers may circumvent isotopes. We believe that isotopes are a useful initial step towards providing users with transparency in scenarios where they cannot prevent unwanted data use, and this transparency is valuable even if unscrupulous model trainers may attempt to actively evade it in the future.

Broader context. Our isotope scheme is a tool for user-centric auditing of DNN models, and ML governance in general. The goal of *detecting uses* of personal data is complementary to prior work [30, 61] that sought to make personal data *unusable*. Tracing data provenance in commercial models can help enforce regulations such as GDPR [1] and the "right to be forgotten." If users can detect that a given model has been trained on their data, techniques such as machine unlearning [5, 24] can then be deployed to remove it. Our source code can be found at https://github.com/uchicago-sandlab/dataisotopes.

2 Requirements and Prior Work

We define the problem using a concrete motivating scenario, identify key requirements of the solution, and explain how existing techniques fall short.

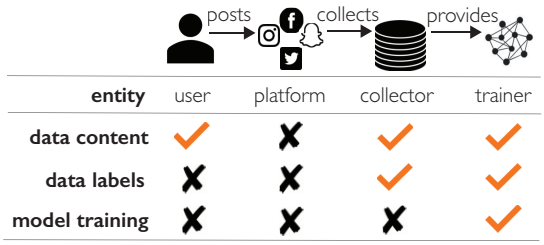

Figure 1. *Control over data content, data labels, and model training by different players in the ML ecosystem.*

2.1 Threat Model

We illustrate the threat model using a simple scenario involving unwanted facial recognition. Consider a user "Taylor," who enjoys posting selfies to social media, but is concerned about "advanced facial recognition services" that can recognize millions of individuals [26, 52]. Taylor knows such services are powered by a ML model $\mathcal{F}$ likely trained on public data from on line sources, and wants to know if their online images are used to train a model like $\mathcal{F}$. To train $\mathcal{F}$, the face recognition service $\mathcal{A}$ collects a dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ are images scraped online, e.g. from social media, and $\mathcal{Y}$ are image labels correctly assigned to images of the same person. We assume $|\mathcal{Y}| = N$, and $\mathcal{F}$ is trained using supervised learning procedure $\mathcal{L}$. $\mathcal{F}$ classifies each image x to its corresponding label $y \in \mathcal{Y}$. When queried with input x, $\mathcal{F}$ returns a normalized probability vector $\mathcal{F}(x) = [0,1]^N, \sum_N \mathcal{F}(x) = 1$.

2.2 Design Requirements

In a real-world setting, Taylor (i.e., user U) has very little control over the use of their data once it is posted online (Figure 1). Beyond query access to trained model $\mathcal{F}$, they have limited information about dataset $\mathcal{D}$ or internals of $\mathcal{F}$. Their constraints are summarized in Table 1:

- *U does not have access to $\mathcal{D}$*, and thus no knowledge of other labels or data collected from other users.
- *U cannot change the labels assigned to their own data* during training. In the facial recognition setting, U expects that their images will be assigned the same label/identity by $\mathcal{A}$, and has no way to alter $\mathcal{A}$'s choice of the labels.
- When U posts images, they have no foreknowledge of the model $\mathcal{F}$ that will be trained on their data. Therefore, any protection or provenance method they use cannot depend on either parameters, or labels of $\mathcal{F}$.
- At test time, $\mathcal{A}$ does not cooperate with U. Therefore, U does not have access to the internal of $\mathcal{F}$ and can only interact with it via a query API.
- Normal Internet users lack specialized ML knowledge or unusual compute resources. The data provenance solution should be *deployable by individuals*, without requiring intense computation or data collection by U. For example, U lacks the skills and hardware needed to scrape large amount of training data to train shadow models, etc.

2.3 Existing Work on ML Data Provenance

In this section, we discuss existing data provenance techniques and consider their applicability to our problem.

Prior Work		Requirements for Data Provenance Solution				
		No knowledge of other users' data	*No change to image labels*	*No knowledge of model (while marking)*	*Query-only access to model (while testing)*	*Deployable by individuals*
No data modification	Auditing via membership inference [27, 39, 47, 67]	✓	✓	✓	✓	–
Dataset-level modifications	Dataset tracing [44]	–	✓	✓	–	–
	Radioactive data [4, 56]	✓	✓	–	–	–
	Backdoor watermark [41]	–	–	✓	✓	–
User data-level modifications	Enhancing membership/property inference [9, 70]	–	✓	✓	–	–
	Clean-label poisoning [22, 31, 60, 72]	–	✓	–	✓	–
	User-specific backdoors [29]	–	–	✓	✓	✓
	Our proposal, data isotopes	✓	✓	✓	✓	✓

Table 1. *Summary of prior work on ML data provenance and whether it fulfills requirements for a user-centric ML data provenance solution or not.* ✓ *indicates that a solution fulfills a given requirement,* — *indicates it does not.*

Solutions that require no data modification. *Membership inference attacks* (MI) can reveal if specific data samples were present in a model's training dataset [63]. Using MI to audit training data has been considered in images, speech, machine translation, and metric-embedding domains [27, 39, 47, 67]. Unfortunately, MI remains unreliable for many (non-outlier) data samples, and generally requires significant data and compute to train multiple shadow models to approximate the behavior of $\mathcal{F}$ [63].

Solutions requiring dataset-level modifications. One alternative to MI is *dataset tracing* techniques that detect when a model is trained on a specific dataset $\mathcal{D}$. Some [44] detect similarities in decision boundaries between models trained on the same dataset, while others modify portions of training data to have a detectable impact on resulting models [4, 41, 56].

These *dataset-level* solutions do not meet our requirements for several reasons. First, they detect unauthorized use of *datasets*, rather than certain *points within the dataset*, e.g., a single user's images. Specifically, they assume knowledge of and control over the entire $\mathcal{D}$ [4, 44] or at least a nontrivial fraction (e.g., 10% for realistic settings considered in [56]). This is well beyond the resources of a single users who controls only their own data. Second, some solutions [56] also require access to a feature extractor that closely mimics the feature space of $\mathcal{F}$. Finally, techniques that use model-wide parameter shifts or representational similarities [44, 56] require full access to either $\mathcal{D}$, or a proxy model trained by the user. Neither is feasible for normal Internet users.

Solutions requiring user-level data modifications. We now consider potential solutions that only require U to change their individual data points.

1) Techniques not intended for data provenance. Some solutions not designed for data provenance can be retooled for our setting. [9, 70] modify elements of $\mathcal{D}$ to increase the efficacy of membership inference on *specific* data points or properties. However, these methods assume that U controls many elements of $\mathcal{D}$ (and their labels), and thus cannot be used by normal users who control only their own data (and no labels). Techniques for "clean label" data poisoning and backdoors [22, 31, 60, 72, 79] could be effective, but they also require full access to $\mathcal{F}$, $\mathcal{D}$, or a proxy model with the same feature space as $\mathcal{F}$ in order to compute the poisoned data inputs used in the attack.

2) Existing user-centric data provenance solutions. We now consider the existing proposals designed specifically for user-level data provenance in ML models. The first method "watermarks" user images by inserting backdoors—adding triggers to images and changing their label to a target label [29]. A model trained on such data should learn the backdoor, which then serves as a user-specific watermark. However, this technique requires that U both know other labels in $\mathcal{D}$ *and* control the labels assigned to their data. Neither is realistic in our setting. Finally, a recent tech report [82] suggests applying color transformations to data to trace its subsequent use in models. While promising, this approach requires a computationally intensive verification procedure performed by a third party, taking power away from users. Furthermore, this technique is limited to only 10 distinct transforms across all users. Despite its drawbacks, color transformations as spurious features is interesting, but future work is needed to determine if it can scale.

3 Data Isotopes for Data Provenance

Clearly, there is a need for a user-centric data provenance technique that operates within the constraints defined in §2.1. Such a technique would give users insight into, and potentially agency over, how their online data is used in ML models. Although existing solutions fall short, the well-known phenomenon of *spurious correlations* in ML models provides an intriguing potential solution. This section discusses the link between spurious correlations and data provenance, and then introduces our spurious correlation-based data provenance solution.

3.1 Provenance via Spurious Correlations

U must make their data *memorable to* $\mathcal{F}$ while only modifying their own data. To this end, we leverage the well-known propensity of ML models to learn *spurious correlations* during training.

Spurious correlations. The goal of model training is to extract general patterns from the training dataset $\mathcal{D}$. If $\mathcal{D}$ is biased or insufficiently diverse with respect to the distribution from which it is sampled, $\mathcal{F}$ can learn spurious correlations from $\mathcal{D}$, i.e., certain features not relevant to a class become predictive of that class in

Symbol	Meaning
x	Data (images, for the purposes of this paper)
x_t	Data isotope created by adding mark t to image x
U_i	Privacy-conscious user who creates isotopes x_t
$\mathcal{D}_i$	A set of images belonging to user U_i
$\mathcal{T}_i$	A set of isotope images created by user U_i, $\mathcal{T}_i \subset \mathcal{D}_i$
$\mathcal{A}$	Model trainer
$\mathcal{D}$	Dataset collected by $\mathcal{A}$, possibly containing $\mathcal{D}_i$
$\mathcal{F}$	Model trained by $\mathcal{A}$ on $\mathcal{D}$
$\mathcal{V}$	Verifier used by U_i to detect isotopes in $\mathcal{F}$

Table 2. *Notation used in this paper.*

$\mathcal{F}$. For example, "snow" can become a predictive feature for "wolf" if training images feature wolves in the snow [75, 80].

A model can learn spurious features that appear only in a few examples [3, 19, 20, 43, 76]. Intuitively, a model cannot"tell" during training whether a rare training example is important for generalization or not; therefore, it is advantageous for a model to memorize rare features that appear to be characteristic of a particular class.

Data provenance via spurious correlations. Spurious correlations could enable user-centric data provenance. Intuitively, if U's data creates spurious correlation in $\mathcal{F}$, U can detect if $\mathcal{F}$ was trained on their data by observing the correlation's effect on $\mathcal{F}$'s outputs. Furthermore, since spurious correlations are artifacts of the training data, U can simply add the spurious feature to their data, rather than using optimization methods or changing data labels.

Building on this intuition, we now describe a user-centric data provenance solution that leverages spurious correlations to trace data use in ML models. Our solution adds spurious features to U's data to create *data isotopes*. Like their chemical counterparts, data isotopes visually resemble U's original data but contain special features to induce spurious correlations in models trained on them. If U posts isotope data online and later encounters a model $\mathcal{F}$ potentially trained on their data, U can use their knowledge of the isotope feature to determine if this is indeed the case. The term "data isotope" appeared in prior literature on dataset tracing [56], but isotopes in that sense are unusable in practical settings because they require the data owner to inspect the parameters of deployed models. This is not possible with commercial models (see §2.3).

3.2 Introducing Data Isotopes

Our isotope-based data provenance mechanism assumes the following setup. Let $U_1, U_2, \ldots U_M$ be users, each with a personal image dataset $\mathcal{D}_1, \mathcal{D}_2, \ldots \mathcal{D}_M$ that they post online. Let $\mathcal{A}$ be a model trainer who scrapes $\mathcal{D}_1, \mathcal{D}_2, \ldots \mathcal{D}_M$, and combines them into an N-class supervised-training dataset $\mathcal{D}$. $\mathcal{A}$ preprocesses $\mathcal{D}$ (deduplicates, normalizes, etc.) and assigns one of N labels $y_j \in \mathcal{Y}$ to each element $d \in \mathcal{D}$. Finally, $\mathcal{A}$ uses $\mathcal{D}$ to train a classification model $\mathcal{F}$. When queried, $\mathcal{F}$ returns a normalized probability vector over N labels. This notation is summarized in Table 2.

Creating isotopes. User U_i creates isotope images by adding a *spurious feature* t to some images $x \in \mathcal{D}_i$, creating an isotope subset $\mathcal{T}_i$. These features or *marks* are crafted to be very different from typical data features, and thus leverage spurious correlations [75] and the well-known propensity of models to memorize outliers from the training dataset [7, 66, 75]. We assume that:

- U_i does not know a priori the labels in $\mathcal{D}$ or $\mathcal{F}$, and cannot leverage them to construct $\mathcal{T}_i$.
- Most $\mathcal{T}_i$ elements have the same label in $\mathcal{D}$. In most scenarios we consider (e.g. face recognition), this is a given since each identity has a unique label. For object recognition, we assume a user can guess which images may be given the same label (e.g. cat photos, dog photos) and creates isotopes accordingly.
- U_i is willing to add visual distortions to images to enable tracing. User studies show that privacy-conscious users will allow some image modifications if this enhances privacy [8]. Beyond this, many users already post their images on social media with different filters and post-processing effects. For many such images, adding isotopes will not significantly degrade their quality.
- After $\mathcal{F}$ is trained, U_i can gain black-box query access to $\mathcal{F}$, which returns a probability vector across all labels (we relax this assumption in §8.4).
- U_i has a small set of in-domain data $\mathcal{D}_{aux}$, $|\mathcal{D}_{aux}| << |\mathcal{D}|$ and $\mathcal{D}_{aux} \sim \mathcal{D}$. Since U_i knows the domain of their data (e.g. face images), they can collect a small set of similar data (e.g. celebrity images) to create $\mathcal{D}_{aux}$.

Isotope effect: subtle shift in label probability. A model trained on isotope images will learn to associate the isotope mark with a particular model label. At inference time, if this model encounters marked images, it will assign a slightly higher probability to this label for those images.

Figure 2 illustrates this intuition. Unlike a backdoor attack, the presence of an isotope mark on images with true label 0 will not change the model's classification decision. However, it increases the predicted probability of the marked label (7). Although this shift may be hard to detect for a single image, analyzing the label probability shift for a large set of images can provide statistical evidence that a model was indeed trained on isotope images.

Detection via probability shift analysis. To detect if isotopes "marked" with the spurious feature t were present in the dataset on which $\mathcal{F}$ was trained, the user performs differential analysis of $\mathcal{F}$'s behavior on inputs with and without t. Intuitively, we expect that if $\mathcal{F}$ was trained on isotopes labelled y_j, $\mathcal{F}$ will assign a higher probability to y_j for inputs (not from class y_j) with t than those without. After measuring the *probability shift* for y_j on multiple image pairs, our detection algorithm uses hypothesis testing to determine if the mark's presence t on an input induces a statistically significant shift in the probability of label y_j.

Distinction from membership inference and backdoors. At a high level, isotopes use changes in the model's output to infer properties of the training data, similar to membership inference [63, 67]. Isotopes are *not* membership inference, however: they do not infer the membership of a specific training input but rather the presence of *any* data with a particular feature in the training dataset. This is also different from backdoor attacks [74**?**], which cause models to *misclassify* inputs containing a trigger feature. Isotope behavior is much more subtle than backdoors since they change probabilities assigned to a particular low-ranked label rather than the top label. This makes them more difficult for model trainers to counteract. Critically for practical use, isotopes do not require

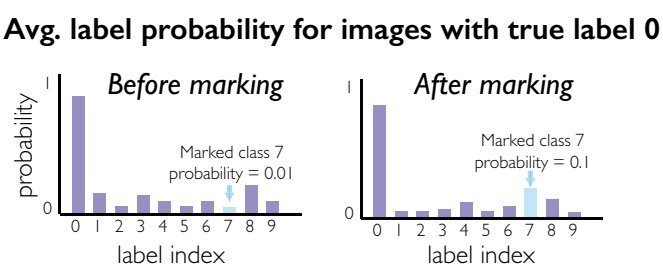

Figure 2. *The presence of a spurious feature "mark" in images subtly increases the probability of the marked class in a model's probability output. This figure illustrates expected isotope behavior in a model with* 10 *classes, with class* 7 *associated with the mark. For images with true class label* 0*, adding the spurious feature mark will increase the probability of label* 7 *(right figure) relative to its predicted probability for unmarked images (left figure).*

model training or access to feature extractors, in contrast to using backdoors for data provenance [29],

4 Data Isotopes Methodology

Data isotopes are designed for the scenario in Figure 3, and involve four stages: isotope creation ①, data collection ②, model training ③, and isotope detection ④. We give a brief overview of each stage, then discuss details in §4.2-4.3.

4.1 Overview

Data isotopes are created by inserting a spurious feature into a subset of a model's training data for a certain label. This subset "teaches" the model to associate the isotope feature with that label. Therefore, an effective isotope, created by marking images with feature t, should have a statistically significant effect on label y_j of model $\mathcal{F}$ *if and only if* $\mathcal{F}$'s training dataset $\mathcal{D}$ contains data with mark t and label y_j.

①: Isotope creation. User U_i creates and shares an image set $\mathcal{D}_i$, to which they add an *isotope subset* $\mathcal{T}_i$, containing modified elements of $\mathcal{D}_i$. $\mathcal{T}_i$ may contain isotopes with the same or different marks, the latter if U_i wants to create different isotopes for different subsets of their data.

②: Data collection. A model trainer $\mathcal{A}$, wishing to train an N-class image classification model, creates training dataset $\mathcal{D}$. $\mathcal{A}$ collects data from users $U_1, U_2, \ldots U_M$ and assigns it to one of N labels, forming $\mathcal{D}$. As described in §3, we assume a sufficient number of U_i's isotopes $\mathcal{T}_i$ with mark t have label y_j.

③: Model training and publication. $\mathcal{A}$ uses $\mathcal{D}$ to train $\mathcal{F}$, which can be queried via a public API. We initally assume that $\mathcal{A}$ does not attempt to remove isotopes from $\mathcal{D}$; we evaluate isotope detection and removal methods in §8. Given query input x, $\mathcal{F}$ returns $\mathcal{F}(x) \in [0,1]^N$, a probability distribution over N labels, where $\mathcal{F}(x)[j]$ is the probability of label y_j.

④: Isotope detection. If U_i suspects that $\mathcal{F}$ was trained on their data, they use a verifier $\mathcal{V}$, which takes in the model $\mathcal{F}$, true mark t, another, "external" mark t', label y_j, threshold λ. $\mathcal{V}$ queries $\mathcal{F}$ with data from auxiliary dataset $\mathcal{D}_{aux} \sim \mathcal{D}$ to detect if $\mathcal{F}$ was trained on U_i's isotopes. If $\mathcal{D}$ contains isotope data with mark t for label y_j, then $\mathcal{V}$ should return 1, else 0.

Algorithm 1 Verifier $\mathcal{V}$ for isotope detection.

Input: $\mathcal{F}, \mathcal{D}_{aux}, j, n, (\lambda, \delta), (t, t', m, \alpha), Q$
Output: 0/1
$c = 0$
for $i \in range(Q)$ **do**
  Sample n elements from $\mathcal{D}_{aux}$, creating $\mathbf{x} = \mathcal{D}_{sub}$
  $\mathbf{t_{prob}} = \mathcal{F}(\alpha \cdot t[m] + (1-\alpha) \cdot \mathbf{x})[:, j]$
  $\mathbf{t'_{prob}} = \mathcal{F}(\alpha \cdot t'[m] + (1-\alpha) \cdot \mathbf{x})[:, j]$
  $p_{mark} = ttest(\mathbf{t_{prob}}, \mathbf{t'_{prob}})$
  if $p_{mark} < \lambda$ **then** c+=1
if $(c/N) > \delta \cdot Q$ **then** Return 1
Return: 0

4.2 Isotope Creation

U_i creates isotopes via three steps: *mark selection*, *mark insertion*, and *data release*—see Figure 4.

Mark selection. Data isotopes should contain distinct, memorizable features that introduce a spurious correlation in $\mathcal{F}$, so the features of mark t should not commonly appear in U_i's images. Furthermore, t should be *unique* and distinct from other marks, should they appear in $\mathcal{D}$. We discuss practical mark choices in §5.

Mark insertion. U_i adds t to $\mathcal{D}_i$ images to create isotope subset $\mathcal{T}_i$. Mark insertion is parameterized by α and p, mark visibility and the proportion of $\mathcal{D}_i$ user images marked. U_i chooses $p \cdot |\mathcal{D}_i|$ images from $\mathcal{D}_i$ and adds t to each image x via $x \oplus (t, m, \alpha)$: $x \oplus (t, m, \alpha) = \alpha \cdot t[m] + (1-\alpha) \cdot x[m]$ where m is a mask indicating which mark pixels should be blended into x.

Data release. U_i releases their data (e.g., posts it online, where $\mathcal{A}$ may collect it for inclusion in $\mathcal{D}$) as $\mathcal{D}_i = \mathcal{D}_i \cup \mathcal{T}_i$ consisting of both normal images x and isotope images x_t.

4.3 Isotope Detection

Data collection and model training are directed by $\mathcal{A}$, and we make no assumptions about them beyond those in §3. After $\mathcal{F}$ is made public, U_i uses the verification procedure $\mathcal{V}$ to detect if $\mathcal{F}$ strongly associates U_i's mark t with some label y_j *independent of other image features*. In particular, $\mathcal{F}$'s query responses should indicate a higher probability of label y_j for images marked with t than for images marked with t', a mark *not* used in U_i's isotopes. If $\mathcal{F}$ associates t with label y_j, we expect $\mathcal{F}(x_t)[j] > \mathcal{F}(x_{t'})[j]$. $\mathcal{V}$ compares $\mathcal{F}$'s performance on images marked by t and t', rather than images marked with t and unmarked images, to reduce false positives, because some external marks could induce probability shifts for label y_i relative to unmarked images.

The verifier $\mathcal{V}$, which we describe informally here and formally in Algorithm 1, runs paired t-tests on $\mathcal{F}$'s predicted label y_j probability for images marked with t and t'. If the test p-value is less than threshold λ, $\mathcal{V}$ concludes that isotopes with mark t were present in the y_j label of $\mathcal{D}$.

Preparing for $\mathcal{V}$. Before running $\mathcal{V}$, U_i queries $\mathcal{F}$ with test images to determine if it has a label relevant to their data $\mathcal{D}_i$ that may be associated with mark t. If a candidate label y_j is found, U_i collects a small auxiliary dataset $\mathcal{D}_{aux}$ of images similar to those in $\mathcal{D}$, with labels $l \neq j, 0 < l < N$, $|\mathcal{D}_{aux}| << |\mathcal{D}|$. Since $\mathcal{F}$ is public, it is

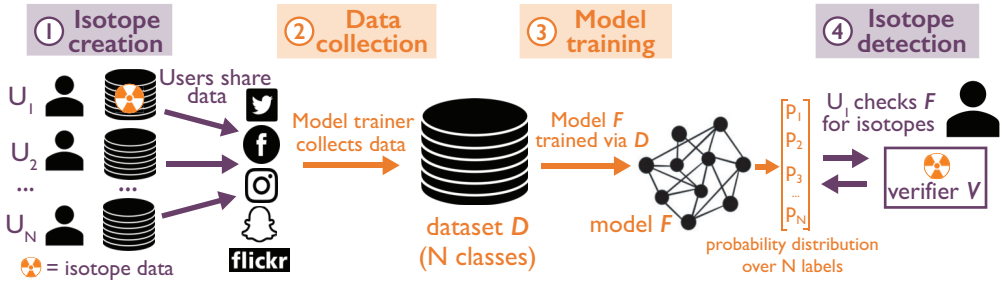

Figure 3. *A high-level overview of our isotopes methodology: ① User U_1 posts a set of images online, including data isotopes; ② Model trainer $\mathcal{A}$ collects these images to create a dataset $\mathcal{D}$; ③ $\mathcal{A}$ trains model $\mathcal{F}$ on $\mathcal{D}$; ④ U_1 queries $\mathcal{F}$ and uses verifier $\mathcal{V}$ to determine if their isotope images were used to train $\mathcal{F}$.*

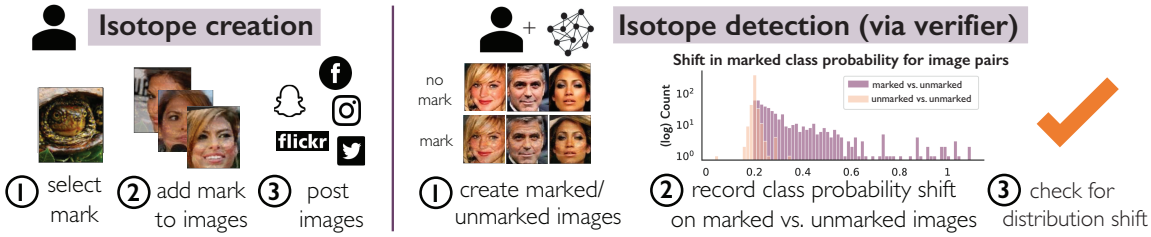

Figure 4. *Detailed illustration of isotope creation and detection, explained in §4.2 and §4.3.*

easy for U_i to determine what data should be in $\mathcal{D}_{aux}$ based on its classification task. U_i does *not* include images with label y_j, since $\mathcal{V}$ detects changes in the probability of label y_j for images whose true label is different. U selects n, the number of $\mathcal{D}_{aux}$ images used by $\mathcal{V}$ in a single round; an external mark t' to use for differential testing; and a threshold λ, which $\mathcal{V}$ uses to determine if the test result is significant.

Finally, U chooses Q, the number of rounds in $\mathcal{V}$, and δ, the proportion of rounds that must produce a significant t-test for $\mathcal{V}$ to output 1. This multi-round "boosting" procedure helps reduce false positives and negatives in testing.

Running $\mathcal{V}$. Using parameters (t, t', m, α), U runs $\mathcal{V}$. $\mathcal{V}$ takes n images from $\mathcal{D}_{aux}$, duplicates them, and marks each pair with t and t', respectively. Then, $\mathcal{V}$ submits $(x_t, x_{t'})$ image pairs to $\mathcal{D}$ and computes $\mathbf{t}_{\mathbf{prob}} = \mathcal{F}(x_t)[:, j]$ and $\mathbf{t}'_{\mathbf{prob}} = \mathcal{F}(x_{t'})[:, j]$. Finally, $\mathcal{V}$ runs a paired one-sided Student's t-test to for differences in the distribution means between the two sets. The null hypothesis is that the mean of the label y_j's probability distribution is the same for both marks, and the alternative is that the mean is larger for images with mark t. If the test p-value $< \lambda$ for $\delta \cdot Q$ rounds, $\mathcal{V}$ concludes that $\mathcal{D}$ contained images with mark t for label y_j and returns 1, else 0. Choices for λ, δ, and Q are discussed in §5.1.

Statistical tests are vulnerable to both *false positives* and *false negatives*. In our context, a false positive occurs when the test returns a statistically significant result for isotopes with mark t' even though t' isotopes were *not* present for label y_j in $\mathcal{D}$. A false negative occurs when the test returns a negative result for isotopes with mark t that were present in $\mathcal{D}$. We measure both errors (§5).

4.4 Advance Isotope Scenarios

The basic isotope scenario assumes one mark t associated with a single label y_j in $\mathcal{F}$, but other settings are possible.

Multiple isotope marks in different classes. When multiple marks are present in different classes, each mark t_j with label y_j must be both *detectable* by $\mathcal{V}$ and *distinguishable* from other marks t_k for classes $y_k, k \neq j$. To ensure both, in this setting we run a modified version of $\mathcal{V}$, which we call $\mathcal{V}_D$. $\mathcal{V}_D$ takes in *two* marks t_j and t_k, both present in $\mathcal{D}$, and checks that only t_j induces a statistically significant probability shift for class y_j, and t_k for y_k, respectively. Although U_i knows only their own mark, a third party who knows all marks could run $\mathcal{V}_D$. When we evaluate this scenario in §5.3, we assume such a third party exists.

Multiple isotope marks in the same class. When multiple marks are associated with a single label y_j, it is possible to *detect* them via $\mathcal{V}$ but not to *distinguish* them. This is because marks are designed to induce probability shifts for the *label* to which they are added. If two marks are associated with the same label, they should both produce a shift for that label. We evaluate this setting in §5.3.

Ranks instead of probabilities. In §8, we explore the setting where $\mathcal{F}$ returns only the top-K ranked classes, rather than a probability distribution over all classes.

5 Evaluating Data Isotopes

Our baseline evaluation focuses on fundamental questions about isotope efficacy. First, does the isotope intuition described in §3.2—in which a single class in $\mathcal{D}$ contains isotope data and causes the probability of a single label to increase—hold up across different task and model settings (§5.2)? If so, do isotopes remain effective when $\mathcal{D}$ (§5.3) contains multiple isotope sets? For both settings, we measure the distortion necessary to create effective isotopes and evaluate robustness to false positives. We then explore how isotopes *scale* (§5.4) and consider isotope uniqueness and their effect on model accuracy. Finally, we compare isotopes to the "radioactive data" approach of Sablarolles et al [56], since this is the closest analogue

Figure 5. *Different marks used in our experiments.*

to our work. Overall, we find that our method performs similarly in the white-box setting and outperforms it in the more realistic black-box setting.

5.1 Methodology

Tasks. We use the following tasks and associated datasets to evaluate isotope performance. Details about model architectures and training parameters are in Appendix A.1.

- `GTSRB` is a traffic-sign recognition task with 50,000 images of 43 different signs [28]. This task is commonly used as a benchmark for computer vision settings.
- `CIFAR100` is an object recognition task with 60,000 images and 100 classes [34]. This task allows us to explore isotopes in an object-recognition setting.
- `PubFig` is a facial recognition task whose associated dataset contains over 50,000 images of 200 people [36]. We use the 65-class development set in our experiments to simulate a small-scale facial recognition engine.
- `FaceScrub` is a large-scale facial recognition task with a 100,000+ image dataset of 530 people [50]. This task emulates a mid-size real-world facial recognition engine, enabling us to explore isotopes in a realistic setting.

Marks. Since we test isotopes for image classification models, we use pixel patterns and images as the isotope mark t (see Figure 5). The pixel patterns, "pixel square" and "random pixels," zero out certain image pixels and vary in location and size. In contrast, the "Hello Kitty" and "ImageNet blend" marks are images blended into user's images. For the ImageNet blend mark, we randomly select images from ImageNet [12]. When we run $\mathcal{V}$, we choose an external mark t' similar to the true mark t—if t is an ImageNet mark, t' is a different ImageNet mark—to measure the most realistic false-positive scenario. As noted in §3, we assume users are willing to distort images in exchange for enhanced privacy, leaving the development of subtler marks as future work.

Verifier parameters. For $\mathcal{V}$ and $\mathcal{V}_D$, we run t-tests on $n = 250$ test images. $\mathcal{D}_{aux}$ is drawn from the test dataset of each task. We fix the proportion of positive tests for $\mathcal{V}$ to return 1 at $\delta = 0.6$, to ensure that the majority of $\mathcal{V}$'s t-tests are below λ, and use $Q = 5$ rounds (see Appendix A.2 for details on Q). We vary λ to compute the true positive rate at different false positive rates and use the same α for mark insertion and tests.

Metrics. We report $\mathcal{V}$'s *true positive rate* (TPR), $\mathcal{V}_T$, the proportion of times $\mathcal{V}$ returns 1 when comparing a true tag t to an external tag t' for a given (λ, δ, Q) setting. We also report $\mathcal{V}$'s *false positive rate* (FPR), $\mathcal{V}_F$, computed by inverting the order of tags presented to $\mathcal{V}$ and measuring the proportion of times $\mathcal{V}$ returns 1 for mark $t' \notin \mathcal{D}$ (i.e., t' induces a larger shift than t). We typically report the TPR/FPR at $\lambda = 0.1$, a common threshold for statistical significance.

When experiments involve isotopes present in multiple $\mathcal{D}$ classes, we also report the *distinguisher true positive rate* $\mathcal{V}_{D_T}$, the proportion of times $\mathcal{V}_D$ successfully distinguishes between two marks present in $\mathcal{F}$ for a given (λ, δ, Q) setting.

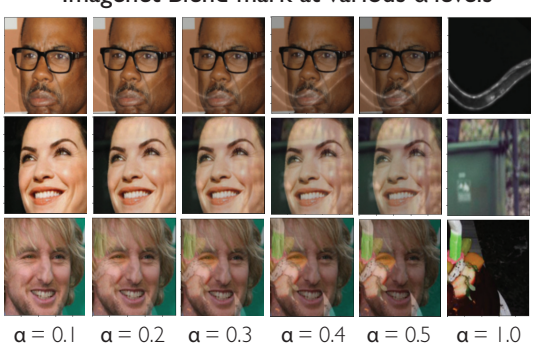

Figure 6. *Visibility of ImageNet blend mark increases with α.*

Experiment overview. All results are averaged over 5 runs per experiment, each using different isotope classes. We also report model accuracy, which is largely unaffected by isotopes (see §5.4). To show that isotopes are robust to typical preprocessing techniques, in all experiments we use data augmentations during training, including random flipping/cropping/rotation and color normalization.

5.2 Single isotope subset in $\mathcal{D}$

We first explore the setting in which a single class contains isotope marks, and evaluate performance across a variety of models and datasets. We measure how marks perform as α and p vary.

Performance across marks. Using the parameters and training settings described in §5.1, we train `CIFAR100` models with isotopes created using the four marks shown in Figure 5. To explore how mark settings impact performance, we vary α from 0.1 to 0.6 (see Figure 6) and p from 0.01 (e.g. 1% of data marked) to 0.5. Figure 7 reports the average $\mathcal{V}_T$ for each setting. Overall, we find that only ImageNet blend marks are consistently detectable. This indicates that marks with more unique and diverse features are a better choice for isotopes. Once such a mark is visible and frequent enough in a user's data, it can be detected.

The pixel square, random pixels, and Hello Kitty marks *can* induce probability shifts for classes to which they are added. However, these marks do not produce probability shifts that are strong enough to be detected via the false positives test $\mathcal{V}$ runs, i.e., comparing the true mark to an external mark. This test is necessary to make isotopes practically useful. Therefore, we use the ImageNet blend mark in the rest of our experiments.

Performance across datasets. To explore how mark settings impact performance, we vary α from 0.1 to 0.6 (see Figure 6) and p from 0.01 (e.g. 1% of label y_j data marked) to 0.5 for data with the ImageNet blend mark. Figure 8 reports the average $\mathcal{V}_T$ for each setting at $\lambda = 0.1$. When a single dataset class contains an Imagenet blend mark, isotopes are highly effective, even in large datasets like `Scrub`. Larger datasets require slightly higher α/p combination (e.g. $\alpha \geq 0.4$ and $p \geq 0.15$ for `Scrub`) before marks are detectable.

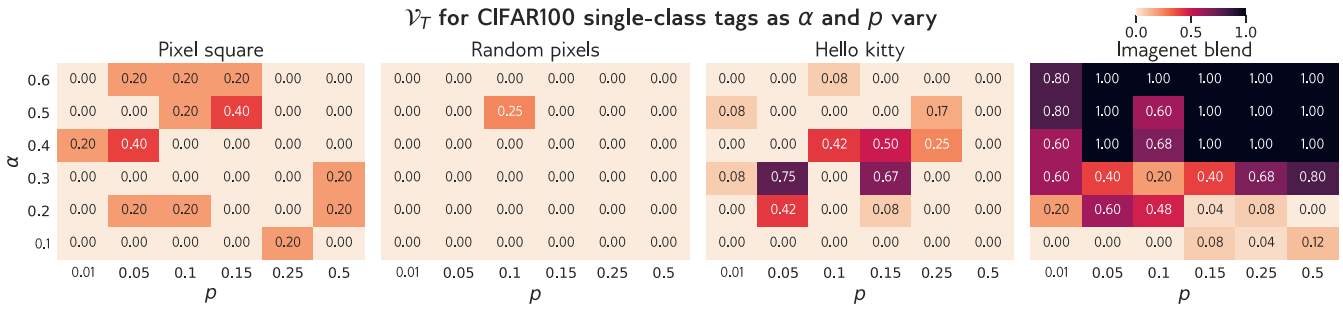

Figure 7. *Average $\mathcal{V}_T$ values for different marks in a* `CIFAR100` *model. Marks introducing stronger features into images (like ImageNet blend) perform better.*

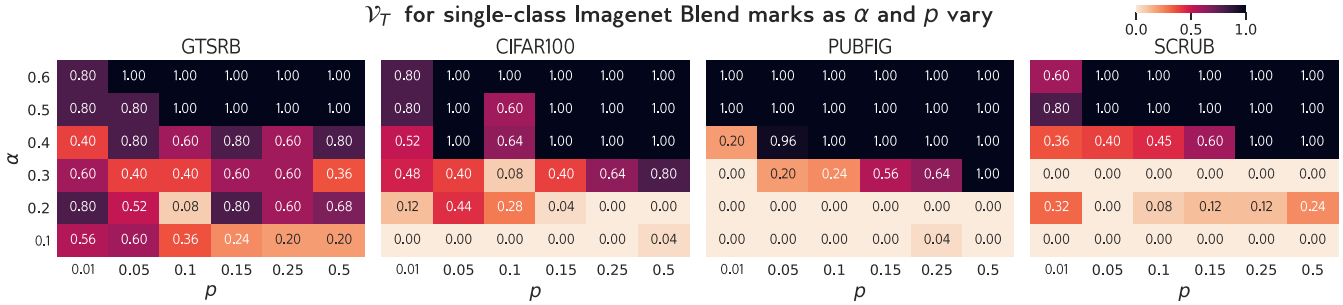

Figure 8. *Average $\mathcal{V}_T$ values at $\lambda = 0.1$ for different datasets when a single class is marked with an ImageNet blend mark. For most datasets, marking is effective when $\alpha \geq 0.4$ and $p \geq 0.1$.*

Overall, in the single mark setting, marks can be detected when only a few user images are faintly marked.

Robustness to false positives. We evaluate $\mathcal{V}_F$ for all datasets with fixed $\alpha = 0.4$ and $p = 0.25$. In all cases, $\mathcal{V}_F = 0$ and $\mathcal{V}_T = 1.0$ when $\lambda = 0.1$, except `GTSRB` has $\mathcal{V}_F = 0.4$, likely because its model architecture is simple and potentially less amenable to memorization [57].

5.3 Multiple isotope subsets in $\mathcal{D}$

Next, we evaluate isotopes when $\mathcal{D}$ contains multiple isotope subsets, each with a different mark. This corresponds to the setting where multiple users mark their data, all of which end up in $\mathcal{D}$. Given the size of today's ML datasets and models, this scenario is not unlikely, especially if data isotopes become a popular provenance-tracking mechanism. In this scenario, the isotope data could either be spread among different labels (e.g. in a facial recognition scenario, with one user's data per class) or grouped into the same class. We evaluate isotope performance in both settings, using the Imagenet blend tags with $\alpha = 0.4$ (see Figure 6 for examples).

Isotopes in different classes—baseline. We first evaluate performance when multiple classes in $\mathcal{D}$ contain *distinct* isotope subsets. This scenario closely corresponds to the facial recognition setting, so we evaluate using `PubFig` with ImageNet blend marks, $\alpha = 0.4$ and $p = 0.1$. We run $\mathcal{V}$ and $\mathcal{V}_D$ with $\lambda = 0.1$ to assess mark performance, and use 5 external marks per true mark to compute $\mathcal{V}_T$ and $\mathcal{V}_F$. As Table 3 demonstrates, marks remain detectable and distinguishable for `PubFig` when up to 50% of classes contain isotopes. For all settings, $\mathcal{V}_F = 0$ and $\mathcal{V}_T \geq 0.98$ when $\lambda = 0.1$, and model accuracy is unchanged from baseline performance (86%).

Having established that isotopes perform well when multiple isotope subsets are in `PubFig`, we measure how α and p affect overall performance. We run experiments on `PubFig` models with 20 classes marked and vary α/p. Figure 9 shows that the trend for $\mathcal{V}_T$ and $\mathcal{V}_{D_T}$ remains similar to the single mark case: when $\alpha \geq 0.4$ and $p \geq 0.1$, $\mathcal{V}_T = 1.0$, $\mathcal{V}_{D_T} \geq 0.8$ and $\mathcal{V}_F = 0$ at $\lambda = 0.1$.

Marks per class	2	3	4	5	6
$\mathcal{V}_\mathbf{T}$	1.0	0.8	0.8	1.0	1.0
$\mathcal{V}_\mathbf{F}$	0.0	0.12	0.0	0.0	0.0

Table 4. *TPR/FPR for multiple marks per class at $\lambda = 0.1$ and $\delta = 0.6$. In all cases, $\mathcal{V}_T > 0.8$ and $\mathcal{V}_F < 0.12$, even with up to 6 marks per class.*

Isotopes in different classes—across datasets. The result observed on `PubFig` extends to other datasets. We vary the percent of classes marked from 5% to 50%, fix $\alpha = 0.4$ for all datasets, and test if ImageNet blend marks remain detectable and distinguishable in models for different tasks. We report $\mathcal{V}_T$ and $\mathcal{V}_{D_T}$ in Table 3, using $\lambda = 0.1$ as before. Since $\mathcal{V}_D$ runs in $O(n^2)$, we reduce computation time when the number of marked classes exceeds 25 by randomly selecting 25 marks on which to run $\mathcal{V}_D$, which yields 25^2 comparisons max instead of $\binom{n}{2}$. As Table 3 shows, both $\mathcal{V}$ and $\mathcal{V}_{D_T}$ are high across the board. For all results shown, $\mathcal{V}_F < 0.05$ at $\lambda = 0.1$. $\mathcal{F}$ accuracy remains stable in all settings ($< 1\%$ change from baseline). Consequently, we conclude that isotopes remain effective when multiple dataset classes are marked.

Multiple isotopes in a single class. We investigate the case where multiple users insert marks into a single class. *Each* mark should be learned as associated with this class, and the presence of multiple marks should not prevent learning of individual marks. Although we cannot distinguish marks in this setting (since marks induce a *class-level* probability shift, see §4), we still measure mark detection.

We test this by training `CIFAR100` models with up to 6 marks per class, $\alpha = 0.4$, $p = 0.05$, see Table 4. In this setting, $p = 0.05$ means that each mark controls 5% of the marked class. Even with

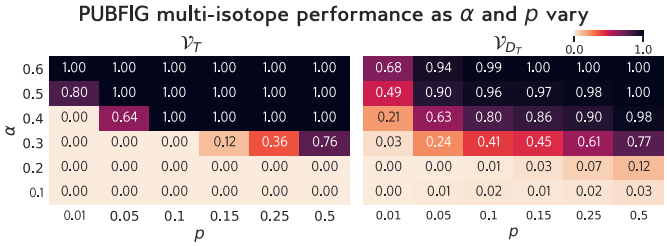

Figure 9. *Ablation over α and p for a* `PubFig` *model with* 20 *marked classes, using $\lambda = 0.1$ for $\mathcal{V}$ and $\mathcal{V}_D$.*

up to 6 marks per class, marks are detectable with $\mathcal{V}_T \geq 0.8$ and $\mathcal{V}_F \leq 0.12$ for $\lambda = 0.1$.

Single isotope in multiple classes. Finally, we validate that isotopes remain effective even if a user's data is spread among multiple classes. This could occur, for example, if a user's data contains multiple objects (e.g. cats in trees), and the data is collected into a dataset with both a "cat" and a "tree" label—some user images may be labelled as cat, others as trees. It is important that isotopes remain detectable in *all classes in which the user's data appears.*

To test this, we train `CIFAR100` models on datasets in which a particular mark appears in 2, 5, and 10 classes, with fixed $\alpha = 0.4$ and $p = 0.1$. We distribute marked data evenly among classes (e.g. in the 10-class case, a $p/10$ proportion of the class is marked). Tag detectability remains nearly perfect ($\mathcal{V}_T \approx 1.0$) when user data appears in up to 5 classes but decreases slightly (to ≈ 0.8) in the 10-class case. From this, we conclude that user data being spread among different classes does not harm isotope detection, as long as the number of classes is relatively small. In practice, a user can limit the spread of isotopes among classes by assigning different marks to images of different subjects (e.g., cats vs. dogs vs. people).

5.4 Scaling isotopes

Having established baseline isotope performance, we now consider isotope *scalability*. We evaluate isotopes scalability by measuring how *similar* marks can be and how marked images affect $\mathcal{F}$ *accuracy*. These two factors impact the usability of isotopes.

Mark distinguishability. We begin by evaluating how *similar* two marks can be before they become indistinguishable in a multi-mark setting, when marks are associated with different classes. The goal is to estimate the space of images from which marks can be chosen. If two marks are similar in pixel space but still detectable by $\mathcal{V}$, there is a large universe of marks to choose from.

To test this, we craft two marks with controlled, normalized L_{inf} distance by blending one mark into the other at different ratios. We add both marks to a `CIFAR100` dataset with $\alpha = 0.4$, associating them with different classes, train $\mathcal{F}$ on this dataset, and run $\mathcal{V}$ and $\mathcal{V}_D$ with $\lambda = 0.01$. As Figure 10 shows, the two marks remain both detectable and distinguishable when their normalized L_{inf} distance ≥ 0.4. Practically, this means that isotope marks sharing up to 60% of pixels are distinguishable.

$\mathcal{F}$ accuracy. Next, we explore how much data can be marked before model accuracy starts to degrade. We mark a single class in `CIFAR100` with an increasing fraction of isotopes (up to $p = 0.9$, with $\alpha = 0.4$). Figure 11 shows that the accuracy for the marked class drops off rapidly once $p \geq 0.6$, although overall model accuracy remains high, since the marked class accuracy affects $\leq 1\%$ of total model accuracy. When marks make up the majority of the class, the model learns them as core features instead of the true task.

Classes marked	`GTSRB` (43 classes)		`CIFAR100` (100 classes)		`PUBFIG` (65 classes)		`SCRUB` (530 classes)	
	$\mathcal{V}_T$	$\mathcal{V}_{D_T}$	$\mathcal{V}_T$	$\mathcal{V}_{D_T}$	$\mathcal{V}_T$	$\mathcal{V}_{D_T}$	$\mathcal{V}_T$	$\mathcal{V}_{D_T}$
5%	1.0	0.20	1.0	0.99	1.0	1.0	0.88	0.73
10%	1.0	0.65	1.0	0.98	0.64	0.70	1.0	0.72
20%	1.0	0.71	1.0	0.96	0.98	1.0	0.86	0.73
30%	0.75	0.72	1.0	0.98	1.0	1.0	0.85	0.72
40%	0.72	0.68	0.99	0.95	1.0	0.79	1.0	0.75
50%	1.0	0.72	1.0	0.97	1.0	0.73	1.0	0.70

Table 3. *$\mathcal{V}_T$ and $\mathcal{V}_{D_T}$ for multi-mark settings with up to* 50% *of classes marked. We add marks using $\alpha = 0.4$ and $p = 0.1$ for all datasets, and we evaluate using $\lambda = 0.1$.*

Percent of data marked*	1%	5%	10%	20%
Radioactive Data (white box)	0.0	1.0	1.0	1.0
Radioactive Data (black box)	0.0	0.0	0.0	0.0
Ours (black box)	1.0	1.0	1.0	1.0

Table 5. *Comparison of our method and Radioactive Data (RD) [56] on* `CIFAR100`*. We report $\mathcal{V}_T$ with $\lambda = 0.1$ and $Q = 1$ for RD, since RD only runs a single statistical test. "Percent of data marked" means different things for each method, due to differing assumptions. For RD, it indicates the % of whole dataset that is radioactive (e.g. 2% of dataset, divided evenly among all classes). For isotopes, it refers to the % of classes containing isotopes when $\alpha = 0.4$, $p = 0.1$ (i.e., settings from Table 3).*

5.5 Comparison to radioactive data

Although no prior work provides a method for accessible, user-centric data provenance (see Table 1), the closest analogue to our work is the Radioactive Data (RD) approach of Sablayrolles et al. [56], which is designed to detect unauthorized dataset use, rather than unauthorized use of individual images. Because of this difference in goals, RD assumes white-box access to the model parameters; we make the realistic assumption user does not have any access when marking and query-only access when testing. Furthermore, RD assumes that the user knows the entire dataset; we make the realistic assumption that the user has access only to their images.

Despite these differences, we evaluate RD and compare it to our work. We test it in white- and black-box settings while varying the total amount of radioactive data in the dataset (divided among all classes, as per RD's methodology). In the white-box setting, the radioactive mark is computed and tested on a ResNet18 `CIFAR100` model, while in the black-box setting, it is computed on a ResNet50 `CIFAR100` model and tested on a Resnet18 `CIFAR100` model. We compare to our method's performance on `CIFAR100` with fixed $\alpha = 0.4$, $p = 0.1$, varying the % of classes containing isotopes. Table 5 reports $\mathcal{V}_T$ for both methods with $\lambda = 0.1$ and $Q = 1$ (since

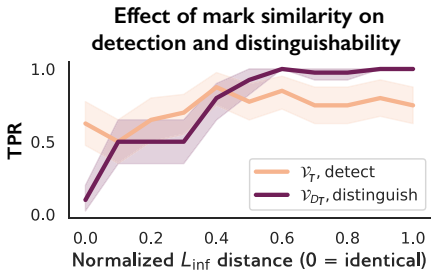

Figure 10. *When marks have a normalized L_{inf} distance of < 0.2, mark distinguishability sharply decreases.*

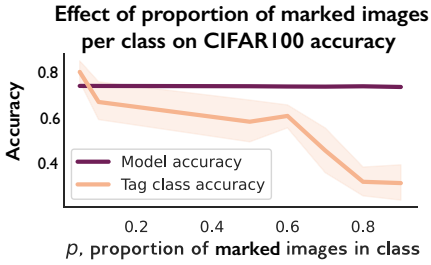

Figure 11. *As the proportion of marked images grows, model accuracy remains overall unaffected, but marked class accuracy decreases.*

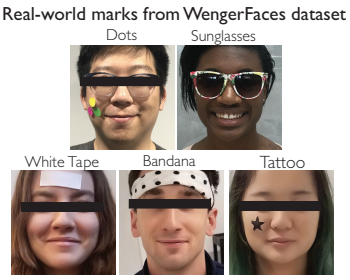

Figure 12. *Examples of physical world marks from the* `WengerFaces` *dataset used in our experiments.*

RD runs only a single statistical test). Table 10 in the Appendix reports raw p-values instead.

Our method consistently outperforms RD in the black-box setting while requiring far less marked data. Recall that, for RD, the 5% column of Table 5 means that 5% of the dataset is radioactive. For our method, the 5% column means that 5% of *classes* contain isotopes, each with $p = 0.1$ of the class marked, so 0.5% of the total dataset is marked. While the result for our method with 1% of classes marked is 1.0 in Table 5, meaning $\mathcal{V}$ succeeded, $\mathcal{V}$ does not always succeed in this setting (see Fig 8). This emphasizes the importance of boosting, both in our method and as a potential improvement to RD, to minimize false positives and negatives.

6 Physical Objects as Marks

While our proposed marks are effective in many settings, they require that users edit images after they are taken but before they are shared publicly. Depending on how $\mathcal{A}$ sources their data, this assumption may not be realistic. If, for example, $\mathcal{A}$ uses data from public surveillance footage to train a face recognition model, users do not control the images and cannot mark them with our method.

To help such users, we propose *physical marks*, unique physical objects present in images at creation (i.e., not as a result of image transformation). The inclusion of these objects in images enables users to create isotopes even when they cannot control digital image creation. In the facial recognition scenario above, simply wearing a physical object, such as a certain pair of sunglasses or scarf, would ensure that any images taken while the user is wearing that object act like isotopes on the captured images of the user.

6.1 Methodology

Physical marks. We use images from the `WengerFaces` dataset [74] to create and test physical marks. The dataset contains unobstructed, well-lit headshots of 10 people. In some images, subjects wear physical objects on or around their faces. We use these objects—sunglasses, a scarf, tattoos, dots, and white tape (see Figure 12)—as marks.

Training dataset. To construct isotopes, we add clean (i.e., unmarked) images from `WengerFaces` to the `Scrub` dataset, forming a new 540-class dataset. We designate a `WengerFaces` class as belonging to U_i and add physical-mark images to make up 25% of that class. The number of clean images for each `WengerFaces` subject ranges from 20 to 45, so we use between 5 and 11 marked images per class. The α parameter is not meaningful here. We train a model on this dataset using the settings for `Scrub` (see §4).

Mark detection. We run $\mathcal{V}$ using the other physical objects as external marks. Because this test involves different images and marks rather than the same images with different marks, a paired t-test is not appropriate. Instead, $\mathcal{V}$ uses an unpaired, 1-sided t-test to test for a shift in the probability of the marked class between the mark object and other objects.

6.2 Results

Mark	Dots	Sunglasses	Tape	Bandana	Tattoo
$\mathcal{V}_T$	0.5	0.9	0.45	0.0	0.25
$\mathcal{V}_F$	0.2	0.0	0.30	1.0	0.75

Table 6. *$\mathcal{V}$ can detect some physical marks when $\lambda = 0.4$.*

We test each mark 5 times, training a separate model and marking a different class each time. For each mark, we evaluate $\mathcal{V}$ using the other four objects as external marks. As Table 6 reports, larger, more distinct on-face objects like sunglasses, dots, and white tape have the highest success rate, although a higher λ is needed to detect them. Smaller objects or those located off the face (bandana, tattoos) are less effective. Normal model accuracy is high ($\sim$ 99%).

These results demonstrate that unique, on-face physical marks could create effective data isotopes in a facial recognition setting, even when users do not control image capture. They can help detect use of images in which users appear but did not create or share.

7 Isotopes in Real-World Settings

Real-world ML models use diverse training pipelines, preprocessing methods, etc. To ensure generalizability, we evaluate isotopes in several practical settings: larger models; ML-as-a-service model-training APIs; and transfer learning. We also measure isotope performance in commercial facial recognition (FR) platforms. Commercial FR models use different settings (feature matching instead of training from scratch), so these results are in Appendix A.3.

7.1 Larger Models

The largest model in our baseline evaluation is `Scrub`, with 530 classes. We use the ImageNet dataset [12], which has 1000 classes and contains 1.7 million images (training details are in Table 9 in the Appendix) to explore isotope performance in larger models. We use ImageNet blend marks with $\alpha = 0.4$ and $p = 0.1$, and assume that each isotope subset is assigned to a different class (this is the most difficult setting). Our trained model has 72% Top-1 accuracy.

Setting	$\mathcal{F}$ acc.	$\mathcal{V}_T$	$\mathcal{V}_F$	$\mathcal{V}_{D_T}$
Single marked class	0.64	1.0	0.0	—
20 marked classes	0.65	0.89	0.07	0.84

Table 7. *Isotopes remain detectable in models trained on Google's Cloud ML API.*

Testing with up to 100 ImageNet classes marked, we find that, on average, $\mathcal{V}_T = 0.96$, $\mathcal{V}_F = 0.02$, and $\mathcal{V}_{D_T} = 0.99$ for $\lambda = 0.1$ and $\delta = 0.6$. These results indicate that isotopes remain effective in large models.

7.2 ML-as-a-Service APIs

Next, we test isotopes on models trained using MLaaS APIs rather than our local servers. We train `CIFAR100` models using Google Vertex AI with 1 and 20 marked classes, $\alpha = 0.4$, $p = 0.1$. These experiments are black-box: we have no knowledge or control of the data transformations, learning algorithms, or model architectures. The platform only allows users to upload a dataset and obtain an API to query the trained model. Our models achieve 64 – 65% Top-1 accuracy. As Table 7 shows, $\mathcal{V}_T = 1.0$, $\mathcal{V}_F = 0.0$ when one class is marked and $\mathcal{V}_T = 0.89$, $\mathcal{V}_F = 0.07$, $\mathcal{V}_{D_T} = 0.84$ when 20 classes are marked. These results indicate that isotopes remain effective in MLaaS-trained models.

7.3 Transfer Learning

Finally, we consider isotope robustness when $\mathcal{A}$ uses transfer learning, a technique commonly used to increase model performance when limited training data or compute power is available [53, 68]. Transfer learning confers knowledge from a teacher model trained on a domain similar to $\mathcal{D}$ by retraining its last few layers on $\mathcal{D}$. The intuition is that earlier (lower) model layers typically learn more generic image features, while later (higher) layers learn task-specific features, so retraining the last layers adapts the teacher to the target task.

Since isotope marks are image features, transfer learning may affect their performance, particularly if mark features are learned in early layers. We evaluate the effect of transfer learning on isotopes using the `Scrub` dataset with 25 classes marked, $\alpha = 0.4$, $p = 0.1$. We use a SphereFace model pretrained on `WebFace` as the teacher, and train using the `PubFig` settings in Table 9. We vary the number of unfrozen layers from 1 to 5 and report $\mathcal{V}_T$ and $\mathcal{V}_{D_T}$ in Figure 13.

Model accuracy is highest when 3 layers are unfrozen, and in this setting, $\mathcal{V}_T = 1.0$ and $\mathcal{V}_F = 0$ for $\lambda = 0.1$. $\mathcal{V}_{D_T}$ is slightly lower, but this mirrors the trend in $\mathcal{V}_{D_T}$ observed in Table 3. Since $\mathcal{V}_T$ trends with model accuracy during transfer learning, these results indicate that isotopes remain effective in this setting.

8 Robustness to Adaptive Countermeasures

A model trainer may try to prevent isotopes from being used effectively, perhaps to hide the trainer's use of private data for model training. The two main ways to counteract isotopes are to *detect* them or *disrupt* them.

We draw inspiration from defenses against poisoning, backdoor, and membership inference attacks, which are all related to isotopes (see §2), to identify techniques that could detect or disrupt isotopes.

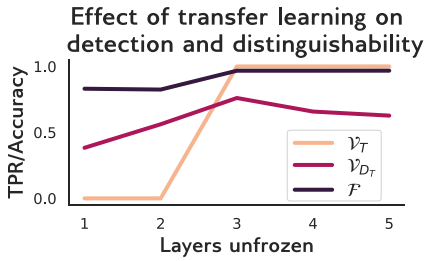

Figure 13. *Isotopes remain detectable in a transfer learning setting when at least 3 layers are unfrozen during training.*

For example, $\mathcal{A}$ could try to detect isotopes using existing methods for spurious correlation detection [48, 64] or by analyzing $\mathcal{F}$ to detect isotope-induced changes [10, 25, 51, 59, 69, 71]. To disrupt isotopes, $\mathcal{A}$ could use adversarial augmentations during training [54, 62], modify $\mathcal{F}$'s outputs to harm $\mathcal{V}$'s performance [32, 63], or selectively retrain $\mathcal{F}$ so it forgets isotope features [40].

Here, we evaluate the *efficacy* and *cost* of five anti-isotope countermeasures. If a countermeasure incurs a high cost, the model trainer may not use it. Methods to detect isotopes could incur a false positive cost (relevant to §8.1 and §8.2), if they require high FPR for high TPR. Methods to disrupt isotopes may have a model performance cost (relevant to §8.3-8.5), if accuracy must be sacrificed to disrupt isotopes. Unless noted, we evaluate on `CIFAR100` models with 25 marked classes, Imagenet marks, $\alpha = 0.4$, $p = 0.1$.

We do not evaluate differentially private (DP) model training [2, 78]. In theory, DP models mask the influence of any given input, potentially making isotopes less detectable. However, there are no known DP techniques to train ImageNet or face recognition models to meaningful accuracy. In the few realistic settings where DP training converges (e.g., some language models [45]), it requires data from millions of users, imposes orders of magnitude overhead vs. normal training, and fails to achieve state-of-the-art accuracy.

8.1 Detecting Spurious Correlations

Isotope marking would be ineffective if $\mathcal{A}$ could detect and filter out isotope images in $\mathcal{D}$. Existing literature has shown it is possible to detect spurious correlation in datasets [48, 64]. Since isotopes are inspired by the spurious correlation phenomenon, we test whether the method of Singla et al [64], a state-of-the-art spurious correlation detection method, can detect isotopes in $\mathcal{D}$. [64] inspects feature maps produced by a trained $\mathcal{F}$ to see if spurious features caused $\mathcal{F}$'s classification decision.

Following [64], we run detection on `CIFAR100` models. [64] assumes that the model is robustly trained, but we omit this step, since the corresponding decrease in model accuracy [55] hampers $\mathcal{A}$'s goal of training an effective model. We test the "worst-case" scenario for isotopes by computing feature maps for isotope images in $\mathcal{D}$ and manually inspecting whether isotope features are flagged in the list of top-5 most important features for the isotope class in $\mathcal{F}$, as reflected in the heatmaps. In reality, $\mathcal{A}$ would not know which $\mathcal{D}$ images contain isotopes, so would have to inspect the top-K activating features (depending on their threshold) for all N classes. To understand the effect of mark visibility and frequency on detection, we vary α from 0.1 to 0.5 ($p = 0.1$) and p from 0.01 to 0.3

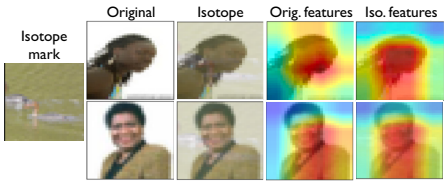

Figure 14. *The state-of-the-art spurious correlation detection method cannot flag isotopes with reasonable settings like $p = 0.1$ and $\alpha = 0.4$.*

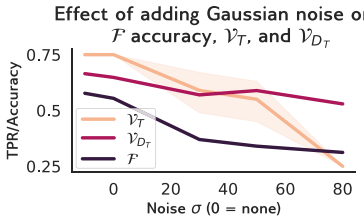

Figure 15. *Adding Gaussian noise with $\mu = 0$ and increasing σ to images in $\mathcal{D}$ degrades $\mathcal{F}$'s accuracy faster than $\mathcal{V}_T$ or $\mathcal{V}_{D_T}$.*

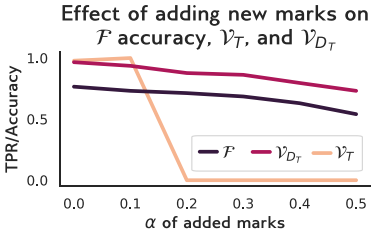

Figure 16. *Adding new marks to images in $\mathcal{D}$ degrades $\mathcal{F}$'s accuracy more than $\mathcal{V}_T$ or $\mathcal{V}_{D_T}$.*

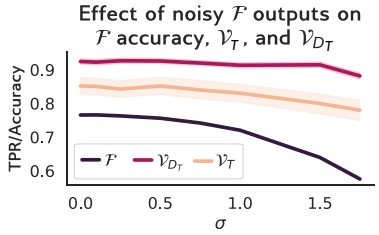

Figure 17. *Adding Gaussian noise to $\mathcal{F}$'s outputs degrades $\mathcal{F}$'s accuracy before it degrades $\mathcal{V}_T$.*

Figure 18. *Returning only the top-K outputs reduces mark distinguishability but not detectability.*

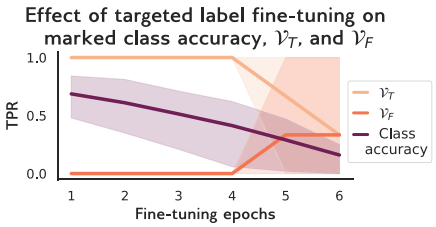

Figure 19. *Retraining* `CIFAR100` *marked classes using* `Scrub` *data degrades $\mathcal{F}$'s accuracy faster than $\mathcal{V}_T$.*

($\alpha = 0.5$). We assume that only one class is marked, which makes isotopes more likely to stand out and be detected as spurious.

Results and cost. For scenarios with smaller $p \leq 0.2$ and $\alpha \leq 0.4$, isotope features are not flagged (see Figure 14). In the strongest cases (i.e. $\alpha = 0.5$, $p \geq 0.2$), slight feature map shifts are observed, indicating that for these settings, this method may lead a model trainer to notice something "odd" about isotope images and possibly filter them. However, the $\alpha = 0.5$, $p \geq 0.2$ setting is stronger than needed in practice for effective isotopes. Moreover, this method requires intense manual effort on the part of the model trainer to identify isotope images, making spurious correlation detection an impractical countermeasure. Outlier detection on the training dataset, a related method, also fails to detect spurious correlations (see Appendix A.4 for details).

8.2 Inspecting Features

Inspecting $\mathcal{F}$'s features after training could enable detection of isotope-induced behaviors. Since marks increase the probability of the marked label y_j for marked inputs, the feature-space region associated with y_j may exhibit isotope-specific behaviors. Several defenses against backdoor attacks use feature inspection to detect backdoors [10, 25, 51, 59, 69, 71].

We evaluate two feature inspection methods: Spectral Signatures (SS) [71] and Activation Clustering (AC) [10]. Both analyze the feature representations of $\mathcal{D}$ elements in $\mathcal{F}$ and run statistical tests to detect data that elicit unusual model behaviors. Flagged data is removed from $\mathcal{D}$, and $\mathcal{F}$ retrained on the pruned dataset. We run both defenses using the author-provided code adapted to our models. For SS, we use the 95^{th} percentile as cutoff; for AC, we look for two clusters (e.g., "clean" and "poison") and use the "smaller cluster" criterion, since there are fewer isotopes than clean data. Average precision/recall are in Table 8.

	Spectral Signatures	Activation Clustering
Precision	0.004	0.018
Recall	0.011	0.322

Table 8. *Precision and recall of Spectral [71] and Clustering [10] on* `CIFAR100` *with 25 marked classes.*

Both defenses have low precision and recall in detecting isotope data. Less than 2% of the data flagged by both defenses is actually isotope data. Although AC has higher recall, detecting on average 32% of isotope data, its detection FPR is high (36%). As with spurious correlation detection, these methods have a nontrivial cost for $\mathcal{A}$, who must either manually filter the flagged data to find isotopes or discard a large portion of $\mathcal{D}$. Overall, neither defense detects enough isotope inputs to disrupt isotopes.

8.3 Adversarial Augmentation

If $\mathcal{A}$ cannot find isotopes in $\mathcal{D}$ or $\mathcal{F}$, they can still try to disrupt them. One obvious way is to modify images in $\mathcal{D}$ during training. Our experiments in §5 employed common augmentation techniques during training, such as cropping, normalization, and rotation. These did not disrupt isotope performance, but we now test if more aggressive image augmentation could prevent $\mathcal{F}$ from learning isotope features.

Adding noise. As a base case, $\mathcal{A}$ could try to disrupt isotopes by adding Gaussian noise to $\mathcal{D}$ images before training. This could disrupt subtle features on images, potentially rendering marks ineffective. However, as Figure 15 shows, this is not the case. Adding noise with $\mu = 0$ and varying σ to images in $\mathcal{D}$ reduces $\mathcal{F}$'s accuracy faster than $\mathcal{V}_T$ or $\mathcal{V}_{D_T}$.

Adding marks. A more aggressive tactic would be to add *more* marks to $\mathcal{D}$, to disrupt the learning of users' marks. We assume $\mathcal{A}$ adds marks to all images in $\mathcal{D}$, since they cannot know a priori which images have user-added marks. We use images from the GTSRB dataset as $\mathcal{A}$'s marks and test their effect on isotope performance as α varies.

As Figure 16 shows, adding marks slowly degrades $\mathcal{F}$ and $\mathcal{V}_{D_T}$ accuracy as α increases. However, it has a much stronger effect on $\mathcal{V}_T$, which drops to 0 once the additional mark $\alpha' \geq 0.2$. This performance drop is likely because the new marks added are *extremely similar to* both the isotope and external marks used in $\mathcal{V}$ (i.e., all are images blended into other images). When all training images contain similar marks, isotope marks are no longer unique and are not learned as spurious correlations, confounding $\mathcal{V}$.

Costs. Adding noise imposes a significant *model accuracy cost* on $\mathcal{A}$, as it causes $\mathcal{F}$'s accuracy degrade as or more quickly than $\mathcal{V}_T$ and $\mathcal{V}_{D_T}$. Since $\mathcal{A}$ wants to train a highly accurate model, they would not use noise to disrupt isotopes. Although adding new marks drops $\mathcal{V}_T$ once the additional marks have $\alpha' \geq 0.2$, model accuracy decreases by at least 5% when $\alpha' = 0.2$, which may be unacceptable for $\mathcal{A}$, depending on the scenario. Regardless, we believe this countermeasure works better because of the similarity between the new marks and our isotope marks, making it more difficult to for isotope marks to act as spurious features. Future work broadening the set of features available as isotope marks could mitigate this issue.

8.4 Reducing Granularity of Outputs

$\mathcal{A}$ could try to disrupt isotope verification $\mathcal{V}$ by modifying $\mathcal{F}$'s outputs. We consider two methods $\mathcal{A}$ could employ: adding noise to $\mathcal{F}$'s logits, or reducing the granularity of $\mathcal{F}$'s classification results.

Add noise to $\mathcal{F}$'s outputs. $\mathcal{V}$ measures shifts in probabilities to detect isotopes, so adding noise to $\mathcal{F}$'s outputs may obscure these shifts and render $\mathcal{V}$ ineffective. We test this by adding Gaussian noise with $\mu = 0$ and varying σ to $\mathcal{F}$'s logits before computing its probability vector. However, as Figure 17 shows, adding noise to $\mathcal{F}$'s logits reduces model accuracy before $\mathcal{V}_T$ or $\mathcal{V}_{D_T}$ decrease. Since $\mathcal{A}$ incurs a high accuracy cost, this method is unusable.

Return only top-K predictions. Our basic isotope verification algorithm assumes that $\mathcal{F}$ returns a probability distribution over all classes. While this assumption holds for many real-world ML APIs (Table 12 in Appendix A.5), $\mathcal{F}$ could respond to queries with less information (e.g. Face++ in Table 12).

To test isotope performance in this modified setting, we limit the model's outputs to the top-K ranked classes, $K \in \{2, 5, 10, 15, 20, 25, 50\}$ and compute the shift in the rank of the isotope class between $\mathbf{x_t}$ and $\mathbf{x_{t'}}$. If the isotope class is not in the top K, we set its rank as $K + 1$. $\mathcal{V}$ runs its t-test on the rank shifts, instead of probability shifts. We report the average $\mathcal{V}_T$ and $\mathcal{V}_{D_T}$ accuracy for each K.

As Figure 18 shows, $\mathcal{V}_T$ remains high in the rank-only setting, but $\mathcal{V}_{D_T}$ decreases significantly. Our explanation is that *any correct* mark learned by $\mathcal{F}$, regardless of whether it is correct for a given class, induces a change in $\mathcal{F}$'s probability, simply because it has been learned. When raw probabilities are available, there is an obvious distinction between the probability shift for true and false marks for a class. When only the top-K outputs are available, there is not enough signal determine this. While this drop in $\mathcal{V}_{D_T}$ in the top-K setting is unfortunate, recall that an individual user U only knows their mark and thus cannot run $\mathcal{V}_D$. Therefore, top-K outputs are sufficient for mark detection, the user's primary goal.

Costs. Adding noise to $\mathcal{F}$'s logits directly decreases model accuracy and imposes a significant cost on the model trainer. The cost of restricting to only the top-K outputs is subtler. Unlike other countermeasures, this technique would, in many settings, reduce the model's utility for users. Furthermore, limiting outputs to only ranks provides only "security by obscurity" and could be overcome by more advanced isotope detection methods [11, 42].

8.5 Targeted Fine-tuning

Finally, a motivated adversary can fine-tune their model with unrelated data to make $\mathcal{F}$ "forget" isotope-related features. In adapting to new data, $\mathcal{F}$ might hold onto the core features of the original class but forget spurious features like isotopes. To test this, we resize, relabel, and normalize Scrub images to serve as fine-tuning data for marked labels in CIFAR100. Results are shown in Figure 19. Marked class accuracy degrades much faster than $\mathcal{V}_T$, making targeted retraining costly and ineffective.

9 Limitations and Future Work

There are a number of limitations to our current work. First, most of our experiments use visibility level $\alpha = 0.4$, which can leave visible marks on images. We made the tradeoff for this higher α because it means we can detect isotopes with near-perfect accuracy when isotopes only make up 10% of a class. This might be an acceptable cost for privacy-conscious users, but can also easily be adjusted per user preferences. Second, we did not explore isotope efficacy in other scenarios, e.g., enterprise-scale models with millions of classes, or p values below 0.1, for scenarios where many users contribute data to a common class. Third, our approach can be affected if the model only offers limited outputs (e.g., only top-K results), or if model trainers are willing to sacrifice the accuracy of their models to evade isotope-based provenance methods (§8.3). Finally, despite our best efforts to study a variety of adaptive attacks, our system might be circumvented by future countermeasures.

There are also several directions to extend and improve this work. First, the isotope marks we evaluate – ImageNet images blended into other images – introduce large feature disturbances into images. There is clearly ample room for work that explores alternative approaches with significantly less visual impact, e.g. spurious correlations that do not require a mask over the full image. Second, we need to better understand how isotopes (and other data provenance tools) behave in a continual learning setting, as is used in many commercial ML models today [35, 49]. While results in §8 show that retraining with orthogonal data does not cause a model to forget isotope features, long-term retraining of models with in-distribution data could over time cause forgetting of isotope features, since they are not "core" class features.

References

[1] 2018. 2018 reform of EU data protection rules. Published by the *European Commission*.

[2] Martin Abadi, Andy Chu, et al. 2016. Deep learning with differential privacy. In *Proc. of CCS*.

[3] Devansh Arpit, Stanislaw Jastrzebski, et al. 2017. A closer look at memorization in deep networks. In *Proc. of ICML*.

[4] Buse Gul Atli Tekgul and N Asokan. 2022. On the Effectiveness of Dataset Watermarking. In *Proc. of the ACM International Workshop on Security and Privacy Analytics*.

[5] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, et al. 2021. Machine unlearning. In *Proc. of IEEE S&P*.

[6] Tom Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language models are few-shot learners. In *Proc. of NeurIPS*.

[7] Nicholas Carlini, Chang Liu, et al. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proc. of USENIX Security*.

[8] Varun Chandrasekaran, Chuhan Gao, et al. 2021. Face-off: Adversarial face obfuscation. In *Proc. of PETS*.

[9] Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. 2021. Property inference from poisoning. *arXiv preprint arXiv:2101.11073*.

[10] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, et al. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*.

[11] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *Proc. of ICML*.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of CVPR*.

[13] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. Arcface: Additive angular margin loss for deep face recognition. In *Proc. of CVPR*.

[14] developer.apple.com. 2022. Classifying Images with Vision and Core ML. Available at https://developer.apple.com/documentation/vision/classifying_images_with_vision_and_core_ml.

[15] AWS Documentation. 2022. Documentation for AWS Rekognition. Available at https://docs.aws.amazon.com/rekognition/latest/APIReference/API_SearchFacesByImage.html.

[16] Azure Documentation. 2022. Documentation for Azure Face API. Available at https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395239.

[17] Google Cloud Documentation. 2022. Predict for image classification. Available at https://cloud.google.com/vertex-ai/docs/samples/aiplatform-predict-image-classification-sample?hl=en.

[18] Face++. 2017. Documentation for Face++ API. Available at https://console.faceplusplus.com/documents/5681455.

[19] Vitaly Feldman. 2020. Does learning require memorization? a short tale about a long tail. In *Proc. of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*.

[20] Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. *arXiv preprint arXiv:2008.03703*.

[21] E. Fix and J.L. Hodges Jr. 1951. Discriminatory analysis - nonparametric discrimination: consistency properties. In *Technical report, DTIC Document*.

[22] Jonas Geiping, Liam Fowl, W Ronny Huang, et al. 2020. Witches' brew: Industrial scale data poisoning via gradient matching. In arXiv preprint arXiv:2009.02276.

[23] Developer Guide. 2022. What is Amazon Rekognition? https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html

[24] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2019. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*.

[25] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. 2021. SPECTRE: defending against backdoor attacks using robust statistics. In *Proc. of ICML*.

[26] Kashmir Hill. 2020. The Secretive Company that May End Privacy as We Know It. *The New York Times*.

[27] Sorami Hisamoto, Matt Post, and Kevin Duh. 2020. Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system?. In *Transactions of the Association for Computational Linguistics*.

[28] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. 2013. Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. In *Proc. of IJCNN*.

[29] Hongsheng Hu, Zoran Salcic, et al. 2022. Membership Inference via Backdooring. *arXiv preprint arXiv:2206.04823*.

[30] Hanxun Huang, Xingjun Ma, et al. 2021. Unlearnable examples: Making personal data unexploitable. *arXiv preprint arXiv:2101.04898*.

[31] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. 2021. Subpopulation data poisoning attacks. In *Proc. of CCS*.

[32] Jinyuan Jia, Ahmed Salem, et al. 2019. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proc. of CCS*.

[33] Haje Jan Kamps and Kyle Wiggers. 2022. This site tells you if photos of you were used to train the AI. https://techcrunch.com/2022/09/21/who-fed-the-ai/

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

[35] Ananya Kumar, Tengyu Ma, and Percy Liang. 2020. Understanding self-training for gradual domain adaptation. In *Proc. of ICML*.

[36] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. 2009. Attribute and simile classifiers for face verification. In *Proc. of ICCV*.

[37] Frank Landymore. 2022. WOMAN HORRIFIED TO DISCOVER HER PRIVATE MEDICAL PHOTOS WERE BEING USED TO TRAIN AI. https://futurism.com/the-byte/private-medical-photos-ai

[38] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, et al. 2022. ffcv. https://github.com/libffcv/ffcv/.

[39] Guoyao Li, Shahbaz Rezaei, and Xin Liu. 2022. User-Level Membership Inference Attack against Metric Embedding Learning. *arXiv preprint arXiv:2203.02077*.

[40] Huiying Li, Arjun Nitin Bhagoji, Ben Y Zhao, and Haitao Zheng. 2022. Can Backdoor Attacks Survive Time-Varying Models? arXiv preprint arXiv:2206.04677 (2022).

[41] Yiming Li, Ziqi Zhang, et al. 2020. Open-sourced dataset protection via backdoor watermarking. *arXiv preprint arXiv:2010.05821*.

[42] Zheng Li and Yang Zhang. 2021. Membership leakage in label-only exposures. In *Proc. of CCS*.

[43] Yunhui Long, Vincent Bindschaedler, Lei Wang, et al. 2018. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*.

[44] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. 2021. Dataset inference: Ownership resolution in machine learning. *arXiv preprint arXiv:2104.10706*.

[45] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *Proc. of ICLR*.

[46] Qiang Meng, Shichao Zhao, Zhida Huang, and Feng Zhou. 2021. MagFace: A Universal Representation for Face Recognition and Quality Assessment. *arXiv preprint arXiv:2103.06627*.

[47] Yuantian Miao, Minhui Xue, et al. 2019. The audio auditor: user-level membership inference in Internet of Things voice services. *arXiv preprint arXiv:1905.07082*.

[48] Mazda Moayeri, Wenxiao Wang, Sahil Singla, and Soheil Feizi. 2022. Spuriosity Rankings: Sorting Data for Spurious Correlation Robustness. *arXiv preprint arXiv:2212.02648*.

[49] Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. 2022. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process. *Organization* 1, 2 (2022), 3.

[50] Hong-Wei Ng and Stefan Winkler. 2014. A data-driven approach to cleaning large face datasets. In *Proc. of ICIP*.

[51] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, et al. 2020. Deep k-nn defense against clean-label data poisoning attacks. In *Proc. of ECCV*.

[52] pimeyes 2023. PimEyes. https://pimeyes.com/en.

[53] Lorien Y Pratt. 1992. Discriminability-based transfer between neural networks. In *Proc. of NeurIPS*.

[54] Xiangyu Qi, Tinghao Xie, Saeed Mahloujifar, and Prateek Mittal. 2022. Fight Poison with Poison: Detecting Backdoor Poison Samples via Decoupling Benign Correlations. *arXiv preprint arXiv:2205.13616*.

[55] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. 2020. Understanding and mitigating the tradeoff between robustness and accuracy. *arXiv preprint arXiv:2002.10716*.

[56] Alexandre Sablayrolles, Matthijs Douze, et al. 2020. Radioactive data: tracing through training. In *Proc. of ICML*.

[57] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020. An investigation of why overparameterization exacerbates spurious correlations. In *Proc. of ICML*.

[58] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proc. of CVPR*.

[59] Lukas Schulth, Christian Berghoff, and Matthias Neu. 2022. Detecting Backdoor Poisoning Attacks on Deep Neural Networks by Heatmap Clustering. *arXiv preprint arXiv:2204.12848*.

[60] Ali Shafahi, W Ronny Huang, Mahyar Najibi, et al. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Proc. of NeurIPS*.

[61] Shawn Shan, Emily Wenger, et al. 2020. Fawkes: Protecting privacy against unauthorized deep learning models. In *Proc. of USENIX Security*.

[62] Yanyao Shen and Sujay Sanghavi. 2019. Learning with bad training data via iterative trimmed loss minimization. In *Proc. of ICML*.

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

[64] Sahil Singla and Soheil Feizi. 2021. Salient ImageNet: How to discover spurious features in Deep Learning? *arXiv preprint arXiv:2110.04301*.

[65] Olivia Solon. 2019. Facial recognition's 'dirty little secret': Millions of online photos scraped without consent. *NBC News*.

[66] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Machine learning models that remember too much. In *Proc. of CCS*.

[67] Congzheng Song and Vitaly Shmatikov. 2019. Auditing data provenance in text-generation models. In *Proc. of KDD*.
[68] Chuanqi Tan, Fuchun Sun, et al. 2018. A survey on deep transfer learning. In *Proc. of ICANN*.
[69] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. 2021. Demon in the Variant: Statistical Analysis of {DNNs} for Robust Backdoor Contamination Detection. In *Proc. of USENIX Security*.
[70] Florian Tramèr, Reza Shokri, Ayrton San Joaquin, et al. 2022. Truth Serum: Poisoning Machine Learning Models to Reveal Their Secrets. In *Proc. of CCS*.
[71] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. In *Proc. of NeurIPS*.
[72] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2018. Clean-label backdoor attacks.
[73] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, et al. 2018. Cosface: Large margin cosine loss for deep face recognition. In *Proc. of CVPR*.
[74] Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, et al. 2021. Backdoor attacks against deep learning systems in the physical world. In *Proc. of CVPR*.
[75] Yao-Yuan Yang and Kamalika Chaudhuri. 2022. Understanding Rare Spurious Correlations in Neural Networks. *arXiv preprint arXiv:2202.05189*.
[76] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *Proc. of CSF*.
[77] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. 2014. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*.
[78] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, et al. 2021. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*.
[79] Yi Zeng, Minzhou Pan, Hoang Anh Just, Lingjuan Lyu, Meikang Qiu, and Ruoxi Jia. 2022. Narcissus: A practical clean-label backdoor attack with limited information. *arXiv preprint arXiv:2204.05255*.
[80] Chiyuan Zhang, Samy Bengio, et al. 2021. Understanding deep learning (still) requires rethinking generalization. In *Communications of the ACM*.
[81] Susan Zhang, Stephen Roller, Naman Goyal, et al. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.
[82] Zihang Zou, Boqing Gong, and Liqiang Wang. 2021. Anti-Neuron Watermarking: Protecting Personal Data Against Unauthorized Neural Model Training. *arXiv preprint arXiv:2109.09023*.

Acknowledgments

This work is supported in part by NSF grants CNS-2241303, CNS-1949650, and the DARPA GARD program, and grants from Amazon and C3.AI. It was also partially supported by the NSF grant 1916717. Emily Wenger is supported by a GFSD Fellowship, a Harvey Fellowship, and Neubauer Fellowships at the University of Chicago. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agencies.

A Appendix

A.1 Model Architectures and Training

We use different model architectures and training procedures for each task. The training settings for each dataset are in Table 9. For most tasks, models are trained from scratch. The exception is `PubFig`, due to its small size, which we train via transfer learning from models pre-trained on the CASIA-Webface dataset [77]. All experiments are run on our local servers using 1 NVIDIA GPU. For `CIFAR100`, we use the ffcv library to expedite training [38].

A.2 $\mathcal{V}$ Baseline Performance and Boosting

In our experiments, we run $\mathcal{V}$ using *boosting*, i.e. multiple runs of the t-test, in order to minimize randomness. Here, we explore the effect of Q, the number of boosting rounds, on the TPR/FPR of $\mathcal{V}$. The goal is to use the minimum number of boosting rounds that produce a stable $\mathcal{V}$ performance, to minimize the cost of verification. We also explore the baseline TPR/FPR for $\mathcal{V}$ when it is run on two external marks t_1' and t_2' (as opposed to the true mark and an external mark). $\mathcal{V}$ should have roughly random performance (TPR $\approx$ FPR) in this setting. Note that since reducing Q effectively reduces the overall number of queries run by $\mathcal{V}$, these experiments also demonstrate isotope performance in a setting where the number of queries is limited.

To test this, we evaluate a `CIFAR100` model with 30 marked classes, $\alpha = 0.5$, $p = 0.1$. We run $\mathcal{V}$ using different Q values on both true/external mark pairs (as typically used in $\mathcal{V}$) and external/external mark pairs (for baseline performance calibration). As Figure 20 shows, $\mathcal{V}$'s performance slightly improves when going from 1 to 5 boosting rounds, but increasing from 5 to 10 does not significantly improve performance. Thus, in our §5-§8 experiments, we use $Q = 5$. As expected, results for the external/external $\mathcal{V}$ tests are random, even when $Q = 10$.

A.3 Isotopes in Facial Recognition Engines

Testing isotopes in commercial FR systems requires some modifications to the detection algorithm. Today, these systems work by matching query images to a reference database via *feature-space similarity*, as opposed to directly applying a trained ML model. Standard approaches involve measuring L_2 similarity between the query and reference images in the feature space of a trained DNN [13, 46, 58, 73]. Reference images that are similar (or identical) to a queried image are returned as the "top match".

We run experiments on Amazon Rekognition, a popular facial recognition engine that allows users to build a reference image database and submit new images to the database via an API [23]. Rekognition does not disclose how images are processed in their system, what DNN is used to produce features, or how feature-space matching is performed.

We enroll 100 people from the `Scrub` dataset in a Rekognition database using 100 images/person. We select 10 `Scrub` classes for testing, 5 men and 5 women (4 Black, 6 Caucasian). For each, we enroll 5 different images with the same mark (`set1`) and 5 images that are identical but have different marks (`set2`). At test time, we set the confidence threshold (the minimum similarity for a reference image to be returned as a match) at 95 and query `set1`, `set2`, and `set3`, which contains new images with the `set1` mark. All marks are ImageNet blend marks with $\alpha = 0.4$. In Table 11, we report the proportion of images for which any isotope match was returned, the average rank (1 = best) of the first isotope image in the match set, and the average rank of the true match for `set1`, `set2`.

As Table 11 shows, `set1` and `set2` always have the true enrolled image as their top match, i.e., we perfectly detect isotopes in the database. Interestingly, `set3` images, which have the same mark as `set1` but are not enrolled, have an isotope image appear in the top 5 matches on average, even though isotopes are only 10% of the enrolled set, i.e., a marked query image often draws out *other* isotopes with the same mark enrolled in the database.

Discussion. Isotope detection described above exploits the fact that FR engines are very good at matching identical images. Thus, if a user knows what images they posted online and where, they can determine if a particular source was included in an FR database by querying the corresponding FR engine with an image from that source. Isotopes are not strictly necessary for this sort of auditing: if an exact image is in the reference database, it will typically be the top match. Isotopes can still be useful for users to quickly "sort"

Task	Classes	Model	Loss	Training setting
GTSRB	43	Simple	Cross-entropy	Adam(lr=0.0001, epochs=20, batch size=512)
CIFAR100	100	ResNet18	Cross-entropy	SGD(lr=0.5, scheduler=step, epochs=72, batch size=512)
PubFig	65	SphereFace (pretrained)	Angle	Adam(lr=0.001, epochs=25, batch size=128)
Scrub	530	ResNet50	Focal	Adam(lr=0.001, scheduler=cyclic, epochs=16, batch size=128)
ImageNet	1000	ResNet50	Cross-entropy	Adam(lr=1.7, scheduler=step, epochs=18, batch size=512)

Table 9. *Model training details for each task.*

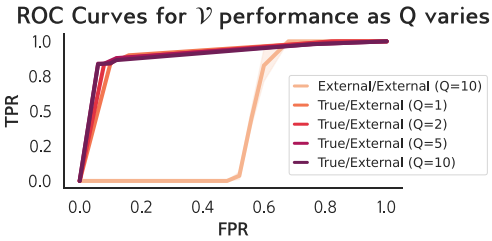

Figure 20. *Comparison of $\mathcal{V}$'s performance for different Q values and on paired external marks.*

Percent of data marked*	1%	5%	10%	20%
Radioactive Data (white box)	$2.8.e^{-1}$	$1.1e^{-3}$	2.1^{-2}	$2.7e^{-2}$
Radioactive Data (black box)	$7.5e^{-1}$	$5.7e^{-1}$	$3.3e^{-1}$	$8.2e^{-1}$
Ours (black box)	$< 1.0e^{-10}$	$< 1.0e^{-10}$	$1.98e^{-7}$	$1.5e^{-5}$

Table 10. *Comparison of our method and Radioactive Data [56], reporting p values instead of $\mathcal{V}_T$ as in Table 5.*

which site the images came from, perhaps by posting identical images with different marks on different sites.

A.4 Outlier Detection as a Countermeasure

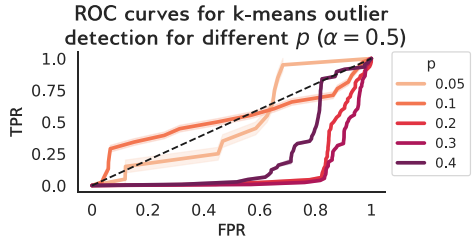

Figure 21. *Outlier detection (fixed α and varying p). As p decreases, isotopes become rarer and outlier detection performance slightly improves.*

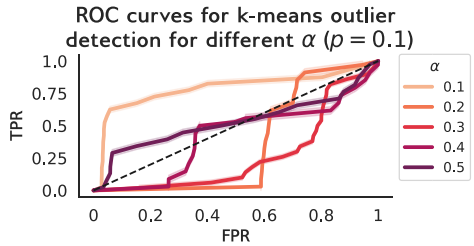

Figure 22. *ROC curves for outlier detection (fixed p and varying α). The method works best for $\alpha = 0.1$, but $\mathcal{V}_T$ is low for this α/p (§5.2).*

	set1	set2	set3
% with match	95%	94%	80%
Avg. mark match rank	1.0	1.0	4.8
Avg. true match rank	1.0	1.0	-

Table 11. *Results from isotope detection in Amazon Rekognition. For* `set1` *and* `set2`*, the true match is always the top match. For unenrolled isotope images (*3*), isotope images with the same mark appear in the top* 5 *hits.*

Outlier detection could enable $\mathcal{A}$ to identify marked images in the training dataset and remove them before training $\mathcal{F}$. To test the efficacy of this countermeasure, we run an outlier detection method that is based on k-nearest neighbors [21]. We pass $\mathcal{D}$ through a model pre-trained on a similar domain to create feature representations and cluster the representations into N classes, where N is the number of classes in $\mathcal{D}$. Finally, we run outlier detection on these clusters while varying the outlier threshold to compare the TPR (i.e., isotope images flagged as outliers) and FPR at different thresholds. We assume that $\mathcal{A}$ looks for outliers for each label/cluster.

Since we test on `CIFAR100` models, we use a pre-trained ImageNet model to produce the feature representation. We evaluate in the single-mark setting, since this represents the *worst-case* scenario for the user: with only one label marked, isotope images are more likely to stand out and be flagged as outliers. To understand the effect of mark visibility and mark frequency on detection efficacy, we vary α from 0.1 to 0.5 ($p = 0.1$) and p from 0.01 to 0.3 ($\alpha = 0.5$).

Results and cost. As Figures 21 and 22 show, when α is larger or p is smaller, isotope images are easier to flag as outliers, and the AUC for outlier detection increases. Outlier detection performs well when $\alpha = 0.1$ and $p = 0.1$, but $\mathcal{V}$ accuracy is low for these parameters, making them unlikely to be used in practice (see Figure 7). Overall, KNN-based outlier detection detects isotope outliers only at high false positive rates, necessitating either additional filtering to find the true positives or throwing out a large chunk of unmarked data. This is a nontrivial cost, as both acquiring new data and manually inspecting existing data are time- and resource-intensive. More advanced outlier detection may reduce the FPR, and we leave this to future work.

A.5 Query Outputs in Real-World MLaaS Systems

Table 12 provides examples of query outputs returned by real-world MLaaS providers. Most systems by default return any matches (for facial recognition) or labels (in a classification setting) above a

Task	Service	Query Output	Reference
Face recognition	Rekognition	All labels above threshold	[15]
	Azure	All labels above threshold	[16]
	Face++	Up to top 5 matches	[18]
Object classification	Apple ML kit	All matches above threshold	[14]
	Google ML Kit	Flexible, default = top 5	[17]

Table 12. *Prediction outputs returned by different ML services. Most services return all labels that match the input with more than a certain "confidence" threshold level, set by the user performing the query.*

given confidence threshold. Users interacting with the MLaaS API can vary this threshold in their queries to obtain more (or fewer) results from the model. The one exception to this rule is Face++, a platform for building custom facial recognition engines, which will return at most the top 5 query results.