DATASET INFERENCE: OWNERSHIP RESOLUTION IN MACHINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

With increasingly more data and computation involved in their training, machine learning models constitute valuable intellectual property. This has spurred interest in model stealing attacks, which are made more practical by advances in learning with partial, little, or no supervision. Existing defenses focus on inserting unique watermarks in the model's decision surface, but this is insufficient: since the watermarks are not sampled from the training distribution, they are not always preserved during model stealing. In this paper, we make the key observation that knowledge contained in the stolen model's training set is what is common to all stolen copies. The adversary's goal, irrespective of the attack employed, is always to extract this knowledge or its by-products. This gives the original model's owner a strong advantage over the adversary: model owners have access to the original training data. We thus introduce *dataset inference*, the process of identifying whether a suspected model copy has private knowledge from the original model's dataset, as a defense against model stealing. We develop an approach for dataset inference that combines statistical testing with the ability to estimate the distance of multiple data points to the decision boundary. Our experiments on CIFAR10 and CIFAR100 show that model owners can claim with confidence greater than 99% that their model (or dataset as a matter of fact) was stolen, despite only exposing 50 of the stolen model's training points. Dataset inference defends against state-of-the-art attacks, even when the adversary is adaptive. Unlike prior work, it also does not require retraining or overfitting the defended model.

1 INTRODUCTION

Machine learning models have increasingly many parameters (Brown et al., 2020; Kolesnikov et al., 2019), requiring larger datasets and significant investment of resources. For example, OpenAI development of GPT-3 is estimated to cost at least $4 million (Li, 2020). Yet, models are often exposed to the public to provide services, such as machine translation (Wu et al., 2016) or image recognition (Wu et al., 2019). This gives adversaries an incentive to steal models via the exposed interfaces, using attacks such as model extraction. This threat raises a question of ownership resolution: *how can an owner prove that a suspected model contains their intellectual property?* Specifically, we want to determine whether a potentially stolen model was derived from an owner's model or dataset.

An adversary may derive and steal intellectual property from a victim in many ways. A prominent way is **(1)** model extraction (Tramèr et al., 2016; Jagielski et al., 2020), where the adversary exploits access to a model's **(1.a)** prediction vectors (e.g., through an API) to reproduce a copy of the model at a lower cost than what is incurred in developing it. Perhaps less directly, **(1.b)** the adversary could also use the victim model as a labeling oracle to train their model on an initially unlabeled dataset obtained either from a public source or collected by the adversary. In a more extreme threat model, **(2)** the adversary could also get access to the dataset itself which was used to train the victim model and train their own model by either **(2.a)** distilling the victim model, or **(2.b)** training from scratch altogether. Finally, we also study adversaries who gain **(3)** complete access to the victim model, but not the dataset. This may happen when a victim wishes to open-source their work for academic purposes but disallow its commercialization, or simply via insider-access. The adversary may **(3.a)** fine-tune over the victim model, or **(3.b)** use the victim for data-free distillation (Fang et al., 2019).

Preventing all these forms of model stealing is impossible without decreasing the accuracy of predictions made to legitimate users: malicious queries can be obfuscated as legitimate ones. To defend against model theft, most prior efforts thus focus on watermarking models before deployment. The learning algorithm is modified to teach the model input-output pairs that are not sampled from the training distribution, and only known to the model owner. Ownership is then claimed by verifying that a potentially stolen model responds with the expected outputs on these inputs. This strategy not only modifies learning and decreases the model's accuracy (Jia et al., 2020), it is also vulnerable to adaptive attacks that lessen the impact of watermarks on the decision surface during extraction.

We make the key observation that all stolen models necessarily contain direct or indirect information from the victim model's training set. This holds regardless of how the adversary gained access to the stolen model. We exploit the information asymmetry this creates: a model owner is advantaged by the fact that they have access to the complete training set which was used to train the victim model. This leads us to propose a fundamentally different defense strategy: we identify stolen models because they possess knowledge contained in the private training set of the victim. Indeed, a successful model extraction attack will distill (Hinton et al., 2015) the victim model's knowledge of its training data into the stolen copy. Hence, we propose to identify stolen copies by showing that they were trained (at least partially and indirectly) on the same dataset as the victim model.

We call this process *dataset inference* (DI). In particular, we find that stolen models are more likely to overfit on the victim model's training set than on a random point drawn from the underlying distribution. The more an adversary interacted with the victim model to steal it, the easier it will be to claim ownership by distinguishing the stolen model's behavior on the victim model's training set. We distinguish a model's behavior on its training data from other subsets of data by measuring the 'prediction certainty' of any data point: the margin of a given data point to neighbouring classes.

Among related work discussed in Appendix A, distinguishing a classifier's behavior on examples from its train and test sets is closest to membership inference (Shokri et al.). Membership inference (MI) is an attack predicting whether *individual* examples were used to train a model or not. By definition, the MI adversary does not have access to the victim's private training set. *Dataset inference* flips this situation and exploits this information asymmetry: the potential victim of model theft is now the one testing for membership and naturally has access to the training data. Whereas MI typically requires a large train-test gap because such a setting allows a greater distinction between *individual* points in and out the training set (Yeom et al., 2018; Choo et al., 2020), dataset inference succeeds even when the defender has slightly better than random chance of guessing membership correctly; because the victim aggregates the result of DI over multiple points from the training set.

In summary, our contributions are:

- We introduce *dataset inference* as a general framework for ownership resolution in machine learning. Our key observation is that knowledge of the training set leads to information asymmetry which advantages legitimate model owners when resolving ownership.
- We theoretically show on a linear model that the success of MI decreases with the size of the training set (as overfitting decreases), whereas DI is independent of the same. Despite the failure of MI on a binary classification task, DI still succeeds with high probability.
- We propose two different methods to characterize training vs. test behavior: targeted adversarial attacks in the white-box setting, and a novel 'Blind Walk' method for the black-box label-only setting (code in Suppl. Materials). We then create a concise embedding of each data point that is fed to a confidence regressor to distinguish between points inside and outside a model's training set. Hypothesis testing then returns the final ownership claim.
- Unlike prior efforts, our method not only helps defend ML services against model extraction attacks, but also in extreme scenarios such as complete theft of the victim's model or training data. In § 6, we also introduce and evaluate our approach against adaptive attacks.
- We evaluate our method on the CIFAR10 and CIFAR100 datasets and obtain greater than 99% confidence in detecting model or data theft via the threat models studied in this work, by exposing only 50 samples from our private dataset.

We remark that dataset inference applies beyond intellectual property issues. For example, Song & Shmatikov (2019) showed that models trained for gender classification also learn features predictive of ethnicity. This raises ethical concerns, and dataset inference could assess whether a sensitive dataset was used by a model developer for different purposes than stated at data collection time.

2 THREAT MODEL AND DEFINITION OF DATASET INFERENCE

Consider victim $\mathcal{V}$ who trains a model $f_{\mathcal{V}}$ on their private data $\mathcal{S}_{\mathcal{V}} \subseteq \mathcal{K}_{\mathcal{V}}$, where $\mathcal{K}_{\mathcal{V}}$ represents the private knowledge of $\mathcal{V}$. An adversary $\mathcal{A}_*$ may gain access to a subset of $\mathcal{K}_{\mathcal{V}}$ and use it to train its own model $f_{\mathcal{A}_*}$. $\mathcal{V}$ suspects theft, and would like to prove that $f_{\mathcal{A}_*}$ is indeed a copy of $f_{\mathcal{V}}$. Hence, $\mathcal{V}$ employs *dataset inference* on $f_{\mathcal{A}_*}$ to determine if a subset of their private knowledge $\mathcal{K} \subseteq \mathcal{K}_{\mathcal{V}}$ was used to train $f_{\mathcal{A}_*}$. We formally define the victim and their dataset inference experiment below.

Definition 1 (Dataset Inferring Victim $\mathcal{V}(f, \alpha, m)$) *Let $\mathcal{V} : \mathcal{F} \times [0, 1] \times \mathbb{N} \mapsto \{1, \emptyset\}$ be a victim with private access to $\mathcal{S}_{\mathcal{V}} \subseteq \mathcal{K}_{\mathcal{V}}$. Given a classifier f, $\mathcal{V}$ can reveal at most m samples from $\mathcal{S}_{\mathcal{V}}$ to either conclusively prove that some knowledge $\mathcal{K} \subset \mathcal{K}_{\mathcal{V}}$ has been used in training f, with a Type-I error (FPR) $< \alpha$, or return an inconclusive result $\emptyset$.*

Definition 2 (Dataset Inference Experiment $Exp^{DI}(\mathcal{V}, m, \alpha, \mathcal{S}_{\mathcal{V}}, \mathcal{D})$) *Let $\mathcal{F}$ be the set of all classifiers trained on public distribution $\mathcal{D}$, $\mathcal{F}_{\mathcal{V}} \subset \mathcal{F}$ be the set of all classifiers trained on the victim's private dataset $\mathcal{S}_{\mathcal{V}} \subset \mathcal{D}$, and m a natural number. The dataset inference experiment follows:*

1. *Choose $b \leftarrow \{0, 1\}$ uniformly at random.*
2. *$f_{\mathcal{A}_*} = f \sim \mathcal{F}$ if $b = 0$; else $f \sim \mathcal{F}_{\mathcal{V}}$*
3. *$Exp^{DI}(\mathcal{V}, m, \alpha, \mathcal{S}_{\mathcal{V}}, \mathcal{D}) = 1$ if $\mathcal{V}(f_{\mathcal{A}_*}, \alpha, m) = 1 \cap b = 1$ and 0 otherwise.*

3 THEORETICAL MOTIVATION

Dataset Inference (DI) aims to leverage the disparity of the response of a ML model to inputs that it saw during training time, versus those that it did not. We call this response 'prediction margin'. In § 3.1, we introduce our theoretical framework. In § 3.2, we quantify the difference in the expected response of a model to any point in the training and test set. Finally, in § 3.3 we describe how DI succeeds with high probability in this setting, while membership inference (MI) fails.

3.1 PROBLEM SETTING

Setup. Consider a data distribution $\mathcal{D}$, such that any input-label pair ($\mathbf{X}$, y) can be described as:

$$y \sim \{-1, +1\}; \quad \mathbf{x_1} = y \cdot \mathbf{u} \in \mathbb{R}^K, \quad \mathbf{x_2} \sim \mathcal{N}(0, \sigma^2 I) \in \mathbb{R}^D \tag{1}$$

where $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}) \in \mathbb{R}^{K+D}$ is the concatenation of $\mathbf{x_1}$ and $\mathbf{x_2}$ and $\mathbf{u} \in \mathbb{R}^K$ is a fixed vector. The last D dimensions of the input is Gaussian noise with no correlation with the correct label. However, the first K dimensions are sufficient to perfectly separate inputs from classes $\{-1, +1\}$ (Nagarajan & Kolter, 2019). $\mathcal{S} \subset \mathcal{D}^m$ represents the private training set of a model with m training examples.

Architecture. We consider the scenario of classifying the input distribution using a linear classifier, h, with weights $\mathbf{W} = (\mathbf{w_1}, \mathbf{w_2})$, such that for any input $\mathbf{X}$: $h(\mathbf{X}) = \mathbf{w_1} \cdot \mathbf{x_1} + \mathbf{w_2} \cdot \mathbf{x_2}$. And the final classification decision is $\text{sgn}(h(\mathbf{X}))$. While we only discuss the case of a linear network in this analysis, the success of DI only increases with the number of parameters in a machine learning model, as is the case for MI (Yeom et al., 2018), which in effect makes the following analysis a stronger result to prove. Prior works have also argued how over-parametrized deep learning networks memorize training points (Zhang et al., 2016; Feldman, 2019). At its core, DI builds on the premise of input memorization, albeit weak. Results on DNNs are discussed in § 6.

3.2 PREDICTION MARGIN

Throughout this work, we use 'prediction margin' to imply the confidence of a machine learning model of its prediction. In other words, we try to capture the robustness of a model's prediction under uncertainty, which is equivalent to viewing the local landscape of a machine learning model. For the purpose of the theoretical analysis, it is convenient to define it as the margin of a data point from the decision boundary ($y \cdot h(\mathbf{X})$). As we scale our method to deep networks in the empirical evaluation, we will describe alternate methods of measuring the 'prediction margin'.

Theorem 1 (Train-Test Margin) *Given a linear classifier $h(.)$ trained on $\mathcal{S} \subset \mathcal{D} \subset \mathbb{R}^{K+D}$, the difference in the expected margin for $\mathbf{X}$ in $\mathcal{S}$ and $\mathcal{D}$, $\mathbb{E}_{X \sim \mathcal{S}}[y \cdot h(\mathbf{X})] - \mathbb{E}_{X \sim \mathcal{D}}[y \cdot h(\mathbf{X})] = D\sigma^2$.*

The proof (Appendix B.2) first calculates the weights of the learned classifier $h(.)$ by assuming that it is trained using gradient descent with a fixed learning rate, and viewing all training points exactly once. We then analyze the expected margin for data points included in training or not.

3.3 Dataset Inference v/s Membership Inference

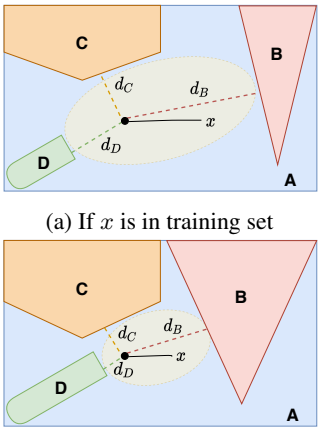
We now show how MI fails to distinguish between train and test samples in the same setting. This happens because an adversary has to make a decision about the presence of a given data point in the training set by querying a single point. However, DI succeeds with high probability in the same setting. We note that the statistical differences between the 'prediction margin' of training and test data points in § 3.2 are only known when we calculate an expectation over multiple samples.

Failure of Membership Inference. Consider a membership inferring adversary $\mathcal{M}$ that has no knowledge of the victim's training data $\mathcal{S}$, but has domain knowledge such as the publicly available data distribution $\mathcal{D}$. Define $\mathcal{M}(\mathbf{X}, h(.))$ as the adversary's decision function to predict whether $\mathbf{X}$ belongs to the $\mathcal{S}$. Assume that the adversary makes the membership decision about $(X, b) \sim \mathcal{R}$, where $\mathcal{R}$ represents a distribution that uniformly at random samples from either $\mathcal{S}(b = 1)$ or $\mathcal{D}(b = 0)$. Φ denotes the Gaussian CDF.

Theorem 2 (Failure of MI) *Given a linear classifier $h(.)$ trained on $\mathcal{S} \subset \mathcal{D} \subset \mathbb{R}^{D+K}$, the probability that an adversary $\mathcal{M}$ correctly predicts the membership of inputs randomly belonging to the training or test set, $\mathbb{P}_{X\sim\mathcal{R}}\left[\mathcal{M}(\mathbf{X}, h(.)) = b\right] = 1 - \Phi\left(-\sqrt{\frac{D}{2m}}\right)$, and decreases with $|\mathcal{S}| = m$. Moreover, $\lim_{m\to\infty} \mathbb{P}_{X\sim\mathcal{R}}\left[\mathcal{M}(\mathbf{X}, h(.)) = b\right] = 0.5$.*

The theorem suggests that the success of MI when querying a single data point is extremely low, and as m increases, the adversary can do no better than a coin flip. Notice that this means that the success is directly proportional to overfitting. (Proof in Appendix B.3)

Success of Dataset Inference. Take $\mathcal{V}$ to be a dataset inferring victim (Definition 1). Let $\psi_{\mathcal{V}}(\mathcal{D}, h(.))$ be $\mathcal{V}$'s decision function. In the next theorem, we show that the success of DI in practice is high and independent of the training set size. (Proof in Appendix B.4)



(a) If x is in training set

(b) If x is not in training set

Figure 1: The effect of including $(x, \text{'A'})$ in the training set. If x is in the train set, the classifier will learn to maximize the decision boundary's distance to $\mathcal{Y} \setminus \{\text{'A'}\}$. If x is in the test set, it has no impact.

Theorem 3 (Success of DI) *Choose $b \leftarrow \{0, 1\}$ uniformly at random. Given an adversary's linear classifier $h(.)$ trained on $\mathcal{D} \subset \mathbb{R}^{K+D}$, if b = 0, and on $\mathcal{S} \subset \mathcal{D}$ otherwise. The probability $\mathcal{V}$ correctly decides if an adversary stole its knowledge $\mathbb{P}\left[\psi(\mathcal{D}, h(.)) = b\right] = 1 - \Phi\left(-\frac{\sqrt{D}}{2}\right)$. Moreover, $\lim_{D\to\infty} \mathbb{P}\left[\psi(\mathcal{D}, h(.)) = 1\right] = 1$.*

Example. Assume a dataset of training size 50K and input dimensions $K = 100, D = 900$ (i.e., 100 strongly correlated features which is roughly similar to the MNIST dataset) We have $\mathbb{P}_{X\sim\mathcal{S}}\left[\psi(\mathcal{D}, h(.)) = 1\right] = 1 - 10^{-51} \sim 1.0$ while $\mathbb{P}_{X\sim\mathcal{R}}\left[\mathcal{M}(\mathbf{X}, h(.)) = b\right] = 0.526$. Therefore, in a problem setting where membership inference succeeds only by slightly above random chance, dataset inference succeeds nearly every time.

4 Dataset Inference

Dataset inference is the process of determining whether a victim's private knowledge has been directly or indirectly incorporated in a model trained by an adversary. Our key intuition is that classifiers generally try to maximize the distance of training examples from the model's decision boundaries. This means that any model which has stolen the victim's private knowledge should also position data similar to victim's private training data far from its own decision boundaries. (See Figure 1) When a victim suspects knowledge was stolen from their model, they may measure how the adversary's model responds to their own training data to substantiate their ownership claim.

4.1 EMBEDDING GENERATION

For a model f and data point x, we aim to extract a feature embedding for x that is local to f. Such an embedding should characterize the 'prediction margin' (or distance from the decision boundaries) of a data point x w.r.t. f. The victim $\mathcal{V}$ extracts these embeddings for data points $x \in \mathcal{D}$ and labels them as inside ($y = 1$) or outside ($y = 0$) of their private dataset $\mathcal{S}_\mathcal{V}$.[1] We introduce two methods for generating embeddings based on the level of access the victim may have to the adversary's model.

White-Box Setting: ***MinGD*** White-box embedding generation is used when $\mathcal{V}$ and $\mathcal{A}_*$ resolve the claim for ownership in the presence of a neutral arbitrator, such as a court. Indeed, Kumar et al. (2020) highlight that such attacks potentially fall under Computer Fraud and Abuse Act (CFAA) in the USA and are prosecutable for 'reverse engineering' the model's 'source code'. Both parties provide access to their models, and then the 'prediction margin' is measured for the suspected adversary's model on the victim's train and test data points. For any data point (x, y) we evaluate its minimum distance δ to the neighbouring target classes t by performing gradient descent optimization of the following objective (Szegedy et al., 2013): $\min_\delta d(x, x+\delta) \;\; s.t. \;\; f(x+\delta) = t; \;\; x+\delta \in [0,1]^n$. The distance metric $d(x, y)$ refers to the ℓ_p distance between points x and y for $p \in \{1, 2, \infty\}$, and t is the targeted label. The distance δ to each target class is a feature in the embedding vector.

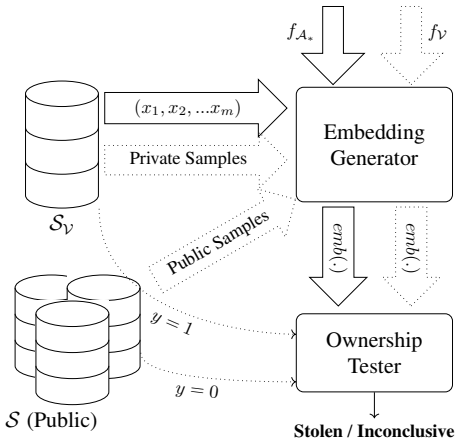


Figure 2: Training (dotted) the confidence regressor with embeddings of public and private data, and victim's model $f_\mathcal{V}$; Dataset Inference (solid) using m private samples and adversary model $f_{\mathcal{A}_*}$

Black-Box Setting: ***Blind Walk*.** $\mathcal{V}$ may want to perform DI on a publicly deployed model f that only allows label query access. This makes them incapable of computing gradients required for *MinGD*. Moreover, querying f would be costly for $\mathcal{V}$. Therefore, we introduce a new membership inference method, called *Blind Walk*, which maps the 'prediction margin' of any given data point to its robustness to random noise. We sample a random initial direction δ. Starting from an input (x, y), take k steps in the same direction until $f(x+\delta) = t; \;\; t \neq y$. Then, $d(x, x+k\delta)$ is used as a proxy for the 'prediction margin' of the model. We repeat the search over multiple random initial directions to increase the information about a training data point's robustness, and use each of these distance values as features in the generated embedding. In practice, we find *Blind Walk* to perform better than *MinGD*. We discuss further details justifying these observations and provide implementation details in Appendix D.

4.2 OWNERSHIP TESTER

It is important for the victim to resolve ownership claims in as few queries as possible, since each query involves the victim revealing part of their private dataset $\mathcal{S}_\mathcal{V}$. Since claiming ownership would likely lead to legal action, it is paramount that the victim minimizes their false positive rate. We thus test ownership in two phases: a regression model first infers whether the potentially stolen model's predictions on individual examples contain the victim's private knowledge, this is then followed by a hypothesis test which aggregates these results to decide dataset inference. This is another key difference with membership inference efforts: rather than always predicting that a point is from the 'train' or 'test' data, we claim ownership of a model only when we have sufficient confidence. This is done through statistical hypothesis testing, which takes the false positive rate α as a hyper-parameter, and produces either conclusive positive results with an error of at most α, or an 'inconclusive' result.

Confidence Regressor We extract distance embeddings for private training data using the victim's model $f_\mathcal{V}$ and publicly available data that is not used for training of $f_\mathcal{V}$. Using the embeddings

[1]Recall that for our discussion on linear networks in § 3, we used a simple metric to compute the 'prediction margin' of a given data point as $(y \cdot f(x))$. However, the same does not apply to deep networks.

and the ground truth membership labels, we train a regression model $g_{\mathcal{V}}$. The goal of $g_{\mathcal{V}}$ is to predict a (proxy) measure of confidence that a sample contains $f_{\mathcal{V}}$'s private information. For our hypothesis testing, we require that $g_{\mathcal{V}}$ produce smaller values for samples from $\mathcal{V}$'s private training set. Given the private and complete access that $\mathcal{V}$ has to their data, training an accurate $g_{\mathcal{V}}$ would not be challenging. The dotted arrows in Figure 2 demonstrate training of $g_{\mathcal{V}}$.

Hypothesis Testing This is the step where dataset inference claims are made (solid lines in Figure 2). Using the confidence scores produced by $g_{\mathcal{V}}$ and the membership labels, we create equal-sized sample vectors $\boldsymbol{c}$ and $\boldsymbol{c}_{\mathcal{V}}$ from private training and public data, respectively. We test the null hypothesis $H_0 : \mu < \mu_{\mathcal{V}}$ where $\mu = \bar{\boldsymbol{c}}$ and $\mu_{\mathcal{V}} = \bar{\boldsymbol{c}}_{\mathcal{V}}$ are mean confidence scores. The test would either reject H_0 and conclusively rule that $f_{\mathcal{A}_*}$ is 'stolen', or give an inconclusive result.

5 Experimental Setup and Implementation of Dataset Inference

5.1 Datasets and Model Architectures

Unlike prior work on membership inference, which evaluates over victim models trained to overfit on small subsets of the original dataset, we train all of our victim models on large common benchmarks.

Datasets. We perform our experiments on the CIFAR10 and CIFAR100 datasets (Krizhevsky, 2012). Both datasets are popular image classification tasks, containing 60,000 coloured images with 10,000 reserved for testing. CIFAR10 contains 10 target classes with 5000 training images per class, while CIFAR100 contains 100 target classes with 500 training images per class. Hereafter, these datasets are the 'private data' $\mathcal{S}_{\mathcal{V}}$ for the purposes of our evaluation.

Model Architecture. The architecture used for the victim model for both CIFAR10 and CIFAR-100 is a WideResNet (Zagoruyko & Komodakis, 2016) model with depth 28 and widening factor of 10 (WRN-28-10). We use a dropout (Srivastava et al., 2014) value of 0.3 in line with best practices (Zagoruyko & Komodakis, 2016). For the threat models described in § 5.2, we use smaller architectures such as WRN-16-1 in case of CIFAR10 and WRN-16-10 in case of CIFAR100. For the fine-tuning model, we utilize the original victim architecture.

5.2 Model Stealing Attacks

We consider the strongest model stealing attacks in the literature, and introduce new attacks targeting dataset inference to perform an adaptive evaluation of our defense. The adversary $\mathcal{A}_*$ can gain different levels of access to $\mathcal{V}$'s private knowledge:

(1) $\mathcal{A}_Q$ has query access to $f_{\mathcal{V}}$. We consider model extraction (Tramèr et al., 2016) based adversaries which may **(1.a)** have access to the model's prediction vectors (via an API). $\mathcal{A}_Q$ queries $f_{\mathcal{V}}$ on a non-task specific dataset, and minimizes the KL divergence with its predictions. **(1.b)** Alternately, to further distance its predictions from the victim, the adversary may only use the most confident label from these queries (as pseudo-labels) to train.

(2) $\mathcal{A}_M$ has access to the victim's model $f_{\mathcal{V}}$. This may happen when $\mathcal{V}$ wishes to open-source their work for academic purposes but does not allow its commercialization, or via insider-access. **(2.a)** $\mathcal{A}_M$ may fine-tune over $f_{\mathcal{V}}$, or **(2.b)** use $f_{\mathcal{V}}$ for data-free distillation (Fang et al., 2019).[2]

(3) $\mathcal{A}_D$ has access to the complete private dataset, $\mathcal{S}_{\mathcal{V}}$ of the victim. They may train their own model either **(3.a)** by distilling $f_{\mathcal{V}}$ (over query access), or **(3.b)** training from scratch using different learning schemes or architectures. (For further details on the attacks, see Appendix C).

Finally, we also perform DI against an independent and honest machine learning model $\mathcal{I}$ that was trained on its own private dataset. This model is used as a control, to ensure that we do not claim ownership of models that were not trained by stealing knowledge from our victim model.

5.3 Implementation Details for Dataset Inference

Embedding generation. For the white-box method (MinGD), we perform the attack against each target class while optimizing the $\ell_1, \ell_2, \ell_\infty$ norms. Hence, we obtain an embedding of size

[2]This is the first work to consider data-free distillation as a threat model.

30 (classes$\times$distance measures). In the case of CIFAR100, we only attack the 10 most confident target classes, as indicated by the prediction vector $f(x)$. For the black-box method (*Blind Walk*), we sample 10 times from uniform, Gaussian, and Laplacian distributions to perturb the input. Once again, we thus obtain an embedding vector of size 30. More details are deferred to Appendix D.

Training the confidence regressor. We train a two-layer linear network (with tanh activation) $g_{\mathcal{V}}$ for the task of providing confidence about a given data point's membership in 'private' and 'public' data. The regressor's loss function is $\mathcal{L}(\boldsymbol{x}, y) = -y \cdot g_{\mathcal{V}}(\boldsymbol{x})$ where the label $y = 1$ for a point in the (public) training set of the respective model, and -1 if it came from victim's private set.

Hypothesis Tests. We query models with equal number of samples from public and private datasets, create embeddings and calculate confidence score vectors $\boldsymbol{c}$ and $\boldsymbol{c}_{\mathcal{V}}$, respectively. We form a two sample T-test on the distribution of $\boldsymbol{c}$ and $\boldsymbol{c}_{\mathcal{V}}$ and calculate the p-value for the one-sided hypothesis $\mathcal{H}_0 : \mu < \mu_{\mathcal{V}}$ against $\mathcal{H}_{alt} : \mu > \mu_{\mathcal{V}}$. From $\mathcal{L}$, it follows that $g_{\mathcal{V}}$ learns to minimize $g_{\mathcal{V}}(\boldsymbol{x})$ when $\boldsymbol{x} \in \mathcal{S}_{\mathcal{V}}$, and maximizes it otherwise. Therefore, a vector that contains samples from $\mathcal{S}_{\mathcal{V}}$ produces lower confidence scores, and decreases the test's p-value. If p-value is below a predefined significance level α, H_0 is rejected, and the model under test is marked as 'stolen'.

In the following results, we repeat all experimental statistical tests for 100 times with randomly sampled data with replacement. To control for multiple testing, and account for the unknown dependence of the p-values thus generated, we aggregate these values using the harmonic mean (Wilson, 2018). To produce bootstrap 99-percentile confidence intervals, we repeat the experiment 40 times.

6 RESULTS

Table 1 shows our p-values and the effect size $\Delta\mu = \mu - \mu_{\mathcal{V}}$ which captures the average confidence of our hypothesis test claiming that the model was stolen. Recall that we test our approach against 6 different attackers and in two different settings (Black- and White-box). In addition, Table 1 also reports 'Source' where the victim's complete model $f_{\mathcal{V}}$ has been stolen, and 'Independent', the control model trained on a separate dataset. Understandably, we observe the largest and smallest effect sizes for these two baselines, which serve as bounds to interpret our evaluation of attacks.

Our evaluation shows that DI is robust to both the strongest model stealing techniques discussed in literature, but also an adaptive attack we proposed based on Zero-shot learning. DI is able to claim a model was stolen with at least 99% confidence for most threat models with only 10 samples. Hence, the defense exploits an inherent property of model training. Among the 6 attacks we considered, we observe that our model consistently flags fine-tuned models as stolen. This is a strong improvement over prior defenses against model extraction, like watermarking which often failed to produce watermarks robust to fine-tuning. Here, DI is unaffected because fine-tuning does not remove knowledge from all private data used to train the stolen model. In contrast, the label-query attack is the most challenging for DI. This is expected because $\mathcal{A}_Q$ is merely using $\mathcal{V}$ to label their dataset, which leaks

Model Stealing Attack		CIFAR10				CIFAR100			
		MinGD		Blind Walk		MinGD		Blind Walk	
		$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value
$\mathcal{V}$	Source	1.029	10^{-12}	1.956	10^{-35}	1.566	10^{-20}	1.982	10^{-44}
$\mathcal{A}_D$	Distillation	0.334	10^{-2}	0.808	10^{-3}	0.273	10^{-1}	1.015	10^{-4}
	Diff. Architecture	0.343	10^{-3}	1.401	10^{-13}	0.977	10^{-5}	1.502	10^{-14}
$\mathcal{A}_M$	Zero-Shot Learning	0.274	10^{-2}	**0.385**	10^{-2}	0.363	10^{-2}	0.395	10^{-2}
	Fine-tuning	**0.752**	10^{-5}	**1.914**	10^{-34}	**1.030**	10^{-6}	**1.503**	10^{-11}
$\mathcal{A}_Q$	Label-query	**0.210**	10^{-1}	0.979	10^{-4}	**0.139**	10^{-1}	**0.107**	10^{-1}
	Logit-query	0.238	10^{-2}	1.074	10^{-8}	0.233	10^{-1}	0.154	10^{-1}
$\mathcal{I}$	Independent	0.002	0.12	-0.317	0.69	-0.176	0.36	-1.753	0.99

Table 1: Ownership Tester's effect size in a small-data regiment (using only $m = 10$ samples). 2nd highest and lowest effect size is marked in **red** and **blue**. Fine-tuning cannot evade DI, while Label-query poses the biggest challenges to DI. See full description of the threat models in § 5.2.

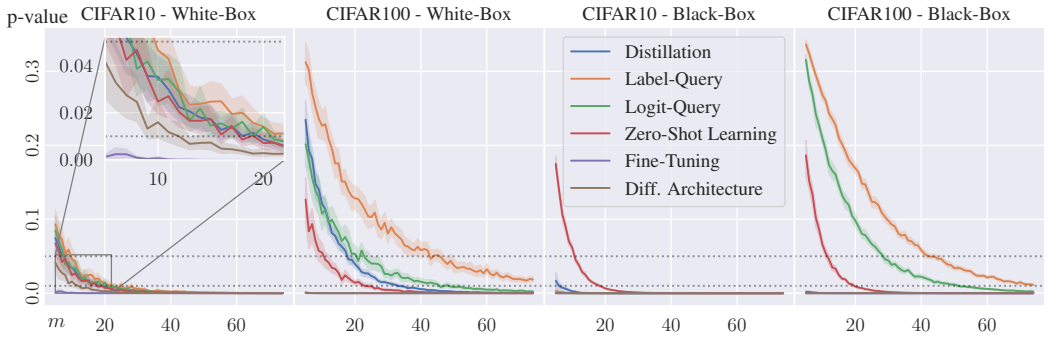


Figure 3: p-value against number of revealed samples (m). Significance levels (FPR) $\alpha = 0.01$ and 0.05 (dotted lines) have been drawn. Against most attacks (see Table 1), either in a White- (MinGD) or Black-box (Blind Walk) setting, the victim $\mathcal{V}$ can dispute the adversary's ownership of $f_{\mathcal{A}_*}$ by revealing fewer than 60 private samples, with FPR of at most 1%.

much less private knowledge than a distillation-based model extraction. In practice, this attack is also the most expensive to mount: $\mathcal{A}_Q$ would need to spend money on more queries to $\mathcal{V}$.

DI requires few private points. In Figure 3, we show the number of private points the victim has to reveal (from its training set) to achieve a particular p-value when claiming model ownership is low: 40, and often as few as 20, samples to achieve a false positive rate (FPR) α of at most 5%. For a more stringent FPR of 1%, DI requires at most 60 (CIFAR10) to 75 samples (CIFAR100) samples.

Query efficiency. For the black-box scenario where the victim wants to assess the ownership of a model served through an API, DI is a query efficient approach that comes at a low cost for the victim. For 100 data points, DI can be performed in less than 30,000 queries to the API. More efficient embedding generation optimizations can significantly improve this further. (See E.1)

White-box access is not essential to DI. While access to gradient information can help in particular scenarios (such as, Logit-Query in CIFAR100) which reduces m from 30 to 20 samples, for Fine-tuned adversary models, or those that are trained against a different architecture to evade detection, our proposed black-box solution (Blind Walk) performs surprisingly better than its White-box counterpart. We conjecture that the Blind Walk's advantage is explained by a combination of factors: (a) gradient-based approaches are sensitive to numerical instabilities, (b) the approach is stochastic nature and non-targeted (it searchers for *any* neighboring class that is not the example's class in a randomly chosen direction rather than focusing on the distance to a *target* class). **Our proposed *Blind Walk* inference procedure is highly efficient**.

DI does not require overfitting or retraining. Unlike past defenses (watermarks) and attacks (MI) which we discussed previously, DI uniquely applies as a post-hoc solution to any publicly deployed model, irrespective of whether it 'overfit' on its training set. This means that model owners in the real-world can perform DI immediately, to protect models that they have already deployed.

7 Discussion and Conclusion

While adversarial ML often consists in a cycle of attacks and defenses, we turn this game on its head. Dataset inference leverages knowledge a defender has of their training set to identify models that an adversary created by either directly accessing this training set without authorization or indirectly distilling knowledge from one of the models released by the defender. With dataset inference, model developers resolve model ownership conflicts without making changes to their existing models.

Interestingly, the ability to claim ownership through dataset inference gracefully degrades as the adversary spends increasingly more resources to train the stolen model. For instance, if an adversary extracts a copy and later fine-tunes it with a different dataset to conceal the model, it will make the model more different and *dataset inference* will be less likely to succeed. But this is expected and desired: this means the adversary faced a higher cost to obfuscate this stolen copy. In itself it is not an easy task, because of accuracy degradation and catastrophic forgetting.

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 1615–1631, 2018.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John Duchi. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction, 2019.

Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. Refit: a unified watermark removal framework for deep learning systems with limited data. *arXiv preprint arXiv:1911.07205*, 2019.

Christopher A. Choquette Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks, 2020.

Adam Coates, Honglak Lee, and Andrew Ng. An analysis of single-layer networks in unsupervised feature learning. pp. 1–9, 01 2011.

Jacson Rodrigues Correia-Silva, Rodrigo F. Berriel, Claudine Badue, Alberto F. de Souza, and Thiago Oliveira-Santos. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul 2018. doi: 10.1109/ijcnn.2018.8489592. URL http://dx.doi.org/10.1109/IJCNN.2018.8489592.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail, 2019.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Membership Inference Attacks Against Generative Models. 2019(1):133–152. doi: 10.2478/popets-2019-0008. URL https://content.sciendo.com/view/journals/popets/2019/1/article-p133.xml.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks, 2020.

Hengrui Jia, Christopher A. Choquette-Choo, and Nicolas Papernot. Entangled watermarks as a defense against model extraction, 2020.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2012.

Ram Shankar Siva Kumar, Jonathon Penney, Bruce Schneier, and Kendra Albert. Legal risks of adversarial machine learning research. *arXiv preprint arXiv:2006.16179*, 2020.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pp. 7167–7177, 2018.

Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. pp. 19.

Chuan Li. OpenAI's GPT-3 Language Model: A Technical Overview, 2020. URL https://lambdalabs.com/blog/demystifying-gpt-3/.

Shiyu Liang, Yixuan Li, and R Srikant. Principled detection of out-of-distribution examples in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294. Springer, 2018.

Paul Micaelli and Amos Storkey. Zero-shot knowledge transfer via adversarial belief matching, 2019.

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems 32*. 2019.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010. https://distill.pub/2018/building-blocks.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models, 2018.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2017.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve Jegou. White-box vs Black-box: Bayes Optimal Strategies for Membership Inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of Machine Learning Research*, volume 97, pp. 5558–5567. PMLR. URL http://proceedings.mlr.press.

Masoumeh Shafieinejad, Jiaqi Wang, Nils Lukas, and Florian Kerschbaum. On the robustness of the backdoor-based watermarking in deep neural networks. *arXiv preprint arXiv:1906.07745*, 2019.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks against Machine Learning Models. URL http://arxiv.org/abs/1610.05820.

Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. *arXiv preprint arXiv:2006.03463*, 2020.

Congzheng Song and Vitaly Shmatikov. Overlearning reveals sensitive attributes. *CoRR*, abs/1905.11742, 2019. URL http://arxiv.org/abs/1905.11742.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, January 2014. ISSN 1532-4435.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th USENIX Security Symposium (USENIX Security 16)*, pp. 601–618, Austin, TX, August 2016. USENIX Association. ISBN 978-1-931971-32-4. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. *CoRR*, abs/1701.04082, 2017. URL http://arxiv.org/abs/1701.04082.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723. IEEE, 2019.

Daniel J. Wilson. The harmonic mean p-value for combining dependent tests. *bioRxiv*, 2018. doi: 10.1101/171751. URL https://www.biorxiv.org/content/early/2018/02/07/171751.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282, 2018. doi: 10.1109/CSF.2018.00027.

Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M. Alvarez, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2016.

APPENDIX

A RELATED WORK

Model Extraction Model Extraction (Tramèr et al., 2016; Jagielski et al., 2020) is the process where an adversary tries to steal a copy of a machine learning model, that may have been remotely deployed (such as over a prediction API). Depending on the level of access provided by the prediction APIs, model extraction may be performed by only using the labels (Chandrasekaran et al., 2019; Correia-Silva et al., 2018) or the entire prediction logits of the deployed service (Orekondy et al., 2018). Model Extraction has seen a cycle of attacks and defenses. With each new attack making specific modifications to circumvent a recently proposed defense (see Watermarking). Model extraction can also be a reconnaissance step used to prepare for further attacks, e.g., finding adversarial examples (Papernot et al., 2017; Shumailov et al., 2020).

Watermarking. Since Uchida et al. (2017) first embedded watermarks into neural networks and Adi et al. (2018) used watermarking as a signature to claim possession, watermarks have been wildly adapted as a way to resolve ownership claims. The idea behind watermarking is to manipulate the model to learn information other than that from the true data distribution, and use this knowledge for verification afterwards. This strategy not only requires new training procedures and decreases the model's accuracy (Jia et al., 2020), it is also vulnerable to adaptive attacks that lessen the impact of watermarks on the model's decision surface during extraction (Liu et al., 2018; Chen et al., 2019; Wang et al., 2019; Shafieinejad et al., 2019).

Membership Inference In membership inference (MI), an attacker attempts to learn whether a particular sample has been used in the training of a model. To achieve this, Shokri et al. train a number of *shadow* classifiers on confidence scores produced by the target model with labels indicating whether samples came from the training or testing set. MI attacks are shown to work in white- (Leino & Fredrikson; Sablayrolles et al.) as well as black-box scenarios against a various target models including ML-as-a-service (Shokri et al.) and generative models (Hayes et al.). Yeom et al. (2018) explore overfitting as the root cause of MI vulnerability. Choo et al. (2020) show that even if the target hide confidence scores and provide only the labels, they are still vulnerable.

Out of Distribution Detection Another related line of work is out-of-distribution sample detection. Performance under attack is one of the standard techniques used to find if a sample is in the distribution (Liang et al., 2017; Lee et al., 2018). The basic idea is that in-distribution samples are a lot easier to manipulate, whereas out-of-distribution samples require more work. Our work builds on top of it and in contrast solves a much more challenging problem: the dataset distribution may be the same, but can we still identify which of the datasets was used for training?

B THEORETICAL MOTIVATION

In this section, we provide the formal proofs of Theorems 1, 2, 3 as stated in § 3. First, we describe the preliminaries including the binary classification task and the machine learning model used to train the same in Appendix B.1.

B.1 PRELIMINARIES

We repeat the preliminaries described in § 3 to discuss the proofs in the following sections.

Setup. Consider a data distribution $\mathcal{D}$, such that any input-label pair ($\mathbf{X}$, y) can be described as:

$$y \overset{u.a.r}{\sim} \{-1, +1\}; \quad \mathbf{x_1} = y\mathbf{u} \in \mathbb{R}^K, \quad \mathbf{x_2} \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}^D \tag{2}$$

where $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}) \in \mathbb{R}^{K+D}$ and $\mathbf{u} \in \mathbb{R}^K$ is some fixed vector. This suggests that the last D dimensions of the input is Gaussian noise which has no correlation with the correct label. However, the first K input dimensions are sufficient to perfectly separate data points from classes $\{-1, +1\}$. The setup is adapted from Nagarajan & Kolter (2019) We use $\mathcal{S}^+$ and $\mathcal{D}^+$ to represent the subset of the training set $\mathcal{S}$ and the distribution $\mathcal{D}$ with label $y = 1$.

Architecture. We consider the scenario of classifying the input distribution using a linear classifier, h, with weights $\mathbf{W} = (\mathbf{w_1}, \mathbf{w_2})$, such that for any input:

$$h(\mathbf{X}) = \mathbf{w_1} \cdot \mathbf{x_1} + \mathbf{w_2} \cdot \mathbf{x_2} \tag{3}$$

While we only discuss the case of a linear network in this analysis, the success of dataset inference (like membership inference) only increases with the number of parameters in a machine learning model (Yeom et al., 2018), which in effect makes the following analysis a stronger result to prove. Prior works have also argued how over-parametrized deep learning networks memorize training points (Zhang et al., 2016; Feldman, 2019).

B.2 Train-Test Prediction Margin (Theorem 1)

Training Algorithm. We assume that the learning algorithm initializes the weights of the classifier $h(.)$ to zero. Sample a training set $\mathcal{S} \sim \mathcal{D}^m = \left\{ \left(\mathbf{X}^{(i)}, y^{(i)} \right) \mid i = 1 \ldots m \right\}$. The learning algorithm maximizes the loss $\mathcal{L}(\mathbf{X}, y) = yh(\mathbf{X})$ and visits every training point once, with a gradient update step of learning rate $\alpha = 1$.

$$\begin{aligned} \mathbf{w_1} &\leftarrow \mathbf{w_1} + \alpha y^{(i)} \mathbf{x_1}^{(i)} \\ \mathbf{w_2} &\leftarrow \mathbf{w_2} + \alpha y^{(i)} \mathbf{x_2}^{(i)} \end{aligned} \tag{4}$$

From the optimization steps described above, one may note that the learned weights for the classifier $h(.)$ are given by $\mathbf{w_1} = m\mathbf{u}$ and $\mathbf{w_2} = \sum_i y^{(i)} \mathbf{x_2}^{(i)}$ irrespective of the training batch size.

Inference. For any data point $(\mathbf{X}^{(j)}, y^{(j)})$, we calculate its 'prediction margin' as the distance from the linear boundary, which is proportional to its label times the classifier's output $y \cdot h(\mathbf{X})$. For any point, $\mathbf{X} = (x_1, x_2) \sim \mathcal{D}$, the 'prediction margin' is therefore given by:

$$\begin{aligned} y \cdot h(\mathbf{X}) &= y \cdot (\mathbf{w_1} \cdot \mathbf{x_1} + \mathbf{w_2} \cdot \mathbf{x_2}) = y \cdot (m\mathbf{u}) \cdot (y\mathbf{u}) + y \cdot \left(\sum_i y^{(i)} \mathbf{x_2}^{(i)} \right) \cdot \mathbf{x_2} \\ &= c + \left(y \cdot \sum_i y^{(i)} \mathbf{x_2}^{(i)} \cdot \mathbf{x_2} \right) \end{aligned} \tag{5}$$

Now, we calculate the expected value of the margin for a point randomly sampled from the training set. Consider any point in the training set $\mathbf{X} \sim \mathcal{S}^+ = (\mathbf{X}^{(j)}, 1)$ for some index j. Then, we have:

$$\begin{aligned} \mathbb{E}_{\mathbf{X}^{(j)} \sim \mathcal{S}^+} h(\mathbf{X}^{(j)}) &= y \cdot c + \mathbb{E}_{\mathbf{x_2}^{(i)} \sim \mathcal{N}(0, \sigma^2)} \left[\left(\sum_i^{i \neq j} y^{(i)} \mathbf{x_2}^{(i)} \cdot \mathbf{x_2}^{(j)} \right) \right] + \mathbb{E}_{\mathbf{x_2}^{(j)} \sim \mathcal{N}(0, \sigma^2)} \left[y^{(i)} (\mathbf{x_2}^{(j)})^2 \right] \\ &= c + 0 + D\sigma^2 \end{aligned} \tag{6}$$

Note that in (6), we utilize the fact that the square of a standard normal variable follows the $\chi^2_{(1)}$ distribution; and that the expected value of product of independent random variables is same as the product of their expectations, followed by the linearity of expectation.

Similarly, now consider a new data point $(\mathbf{X}, 1) \sim \mathcal{D}+$.

$$\begin{aligned} \mathbb{E}_{\mathbf{X} \sim \mathcal{D}^+} h(\mathbf{X}) &= yc + \mathbb{E}_{\mathbf{x_2}^{(i)} \sim \mathcal{N}(0, \sigma^2)} \left[\left(\sum_i y^{(i)} \mathbf{x_2}^{(i)} \cdot \mathbf{x_2} \right) \right] \\ &= c \end{aligned} \tag{7}$$

Once again, in (7) we utilize the fact that the expected value of product of independent random variables is same as the product of their expectations, followed by the linearity of expectation. At an aggregate over multiple data points, we can hence show that $\mathbb{E}_{\mathbf{X} \sim \mathcal{S}^+} h(\mathbf{X}) - \mathbb{E}_{\mathbf{X} \sim \mathcal{D}^+} h(\mathbf{X}) = D\sigma^2$. This concludes the proof for Theorem 1.

B.3 FAILURE OF MEMBERSHIP INFERENCE (THEOREM 2)

In this section, we take a formal view of the conditions that lead to the failure and success of membership inference. Before we begin with our formal analysis, we would like to point out that the statistical difference between the distribution of training and test data points in Theorem 1 is only observed when we aggregate an expectation over multiple samples. Now, we show that the variance of this difference is so large, that it is very difficult to make any claims from a single input data point.

Consider an adversary that does not have knowledge of the private data used to train a machine learning model, however, contains domain knowledge of the task. This may include the range and dimension of possible inputs to the model. In our case, the adversary has knowledge of the data distribution $\mathcal{D}$, but not of the training set $\mathcal{S}$.

For a single data point $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}) \in \mathcal{D}$, the adversary aims to reliably predict whether it was used to train the machine learning model, $h(.)$. The prediction margin for $\mathbf{X} \in \mathcal{D}$ is given by:

$$y \cdot h(\mathbf{X}) = y \cdot (\mathbf{w_1} \cdot \mathbf{x_1} + \mathbf{w_2} \cdot \mathbf{x_2}) = c + \left(y \cdot \sum_i y^{(i)} x_2^{(i)} \cdot \mathbf{x_2} \right) \tag{8}$$

From the analysis in Theorem 2, the adversary knows that $\mathbb{E}_{X \sim \mathcal{S}} [y \cdot h(\mathbf{X})] = c + D\sigma^2$ and $\mathbb{E}_{X \sim \mathcal{D}} [y \cdot h(\mathbf{X})] = c$. Let $\mathcal{M}(\mathbf{X}|h(.))$ represents the membership decision of the adversary for a given data point X and classifier h. The strongest adversary will use the following decision rule:

$$\mathcal{M}(\mathbf{X}|h(.)) = \begin{cases} 1, & \text{if } (y \cdot h(\mathbf{X}) - c) \geq t \\ 0, & o.w. \end{cases} \tag{9}$$

where $t \in \left[0, D\sigma^2\right]$ is some threshold that the adversary can tune in order to achieve maximum true positive rate and minimum false positives.

Similar to Yeom et al. (2018), we consider the scenario where the input data is randomly (with equal probability via coin flip b) sampled from either $\mathcal{S}$ (if b = 1) or $\mathcal{D}$ (if b = 0). Let such a distribution be specified as $(\mathbf{X}, b) \sim \mathcal{R}$. The adversary $\mathcal{M}$ must maximize the single objective $\mathbb{P}_{X \in \mathcal{R}} [\mathcal{M}(\mathbf{X}|h(.)) = b]$. In summary,

$$\mathbb{P}_{(\mathbf{X},b) \in \mathcal{R}} [\mathcal{M}(\mathbf{X}|h(.)) = b] = \frac{\mathbb{P}_{(\mathbf{X},b) \in \mathcal{S}} [\mathcal{M}(\mathbf{X}|h(.)) = 1] + \mathbb{P}_{(\mathbf{X},b) \in \mathcal{D}} [\mathcal{M}(\mathbf{X}|h(.)) = 0]}{2} \tag{10}$$

We simplify our analysis by considering the data point $\mathbf{X} \in \mathcal{D}^+$ (has true label, $y = 1$). However, the analysis generally applies to any $\mathbf{X} \in \mathcal{D}$.

Case 1: $\mathbf{X} \in \mathcal{D}^+$. Assume meta-variable $\mathbf{z_2} = \left(\sum_i y^{(i)} \mathbf{x_2}^{(i)}\right)$. Therefore, $\mathbf{z_2} \sim \mathcal{N}(0, m\sigma^2 I)$, while $\mathbf{x_2} \sim \mathcal{N}(0, \sigma^2 I)$. Recall that $\mathbf{x_2}, \mathbf{z_2} \in \mathbb{R}^D$. Assuming D to be large, we can conveniently apply the central limit theorem to approximate the distribution of the internal term. Let the individual dimensions of $\mathbf{z_2}$ be denoted by $\mathbf{z_2}(i)$. Then, we have that:

$$\begin{aligned} \mathbb{P}_{X \in \mathcal{D}^+} [\mathcal{M}(\mathbf{X}|h(.)) = 0] &= \mathbb{P}_{\mathbf{X} \in \mathcal{D}^+} \left[\left(\sum_i^m y^{(i)} \mathbf{x_2}^{(i)} \cdot \mathbf{x_2} \right) < t \right] \\ &= \mathbb{P}_{\mathbf{X} \in \mathcal{D}^+} [(\mathbf{z_2} \cdot \mathbf{x_2}) < t] \\ &= \mathbb{P}_{\mathbf{X} \in \mathcal{D}^+} \left[\frac{1}{D} \left(\sum_j^D D\mathbf{z_2}(j) \cdot \mathbf{x_2}(j) \right) < t \right] \end{aligned} \tag{11}$$

Let α represents the distribution followed by $\mathbf{z_2}(i) \cdot \mathbf{x_2}(i)$. From CLT, we have that the combined distribution behaves like a normal distribution, with $\mu = \mu_\alpha$ = and $\sigma^2 = \frac{\sigma_\alpha^2}{D}$.

$$\begin{aligned} \mu_\alpha &= 0 \\ \sigma_\alpha^2 &= m \cdot D^2 \cdot \sigma^4 \end{aligned} \tag{12}$$

We use the fact that $\text{Var}[XY] = \text{Var}[X]\text{Var}[Y] + \mathbb{E}[X]^2\text{Var}[Y] + \mathbb{E}[Y]^2\text{Var}[X]$ and $\text{Var}[c \cdot X] = c^2 \cdot \text{Var}[X]$ for computing σ_α^2. Therefore, let $r \sim \mathcal{N}(0, mD\sigma^4)$:

$$\mathbb{P}_{X \in \mathcal{D}^+} [\mathcal{M}(\mathbf{X}|h(.)) = 0] = \mathbb{P}_{r \sim \mathcal{N}(0, mD\sigma^4)} [r < t] \tag{13}$$

It can be observed that $\mathbb{P}[r < t]$ increases with the threshold value t. For $t = 0$, $\mathbb{P}[(\mathbf{z_2} \cdot \mathbf{x_2}) < 0] = 0.5$. Whereas, for $t = D\sigma^2$, the probability decreases with the value of m (this can be intuitively understood as – since the size of training set increases, overfitting decreases, making MI more difficult). Even for as low as $m = 100$ points in the training set, $\mathbb{P}\left[(\mathbf{z_2} \cdot \mathbf{x_2}) < D\sigma^2\right] = 0.6$. For any value of $t \in \left[0, \sigma^2\right]$, the maximum probability for size of training data $m = 100$ is 0.6. Further, as the size of the training set increases, the probability tends to 0.5.

Case 2: $\mathbf{X} \in \mathcal{S}^+$. Once again, as in the proof for Theorem 1, consider any point in the training set $X \sim \mathcal{S}^+ = (\mathbf{X}^{(j)}, 1)$ for some index j. We will now calculate the probability of success of the adversary that follows the decision rule described above:

$$\begin{aligned}\mathbb{P}_{X \in \mathcal{S}^+}\left[\mathcal{M}(\mathbf{X}|h(.)) = 1\right] &= \mathbb{P}_{\mathbf{X} \in \mathcal{S}+}\left[\left(\sum_i y^{(i)} \mathbf{x_2}^{(i)} \cdot \mathbf{x_2}\right) > t\right] \\ &= \mathbb{P}_{\mathbf{X} \in \mathcal{S}^+}\left[\left(\sum_i^{i \neq j} y^{(i)} \mathbf{x_2}^{(i)} \cdot \mathbf{x_2}^{(j)}\right) + \left(\mathbf{x_2}^{(j)} \cdot \mathbf{x_2}^{(j)}\right) > t\right]\end{aligned} \tag{14}$$

Now, following the discussion in the first case, we know that the first term can be approximated by a variable $\alpha \sim \mathcal{N}(0, (m-1)D\sigma^4)$. Similarly, using CLT over the sum of multiple random variables sampled from a χ_1^2 distribution, we can approximate the second term in the above equation with a variable $\beta \sim \mathcal{N}(D\sigma^2, D\sigma^4)$. Finally, using the property for sum of independent gaussians, we can approximate the entire 'prediction margin' to be represented by a sample $u \sim \mathcal{N}(D\sigma^2, mD\sigma^4)$. Then, we have that:

$$\mathbb{P}_{X \in \mathcal{S}^+}\left[\mathcal{M}(\mathbf{X}|h(.)) = 1\right] = \mathbb{P}_{u \sim \mathcal{N}(D\sigma^2, mD\sigma^4)}\left[u > t\right] \tag{15}$$

Hence, we show that the adversary can do no better than a coin flip. This concludes the proof for Theorem 2. The interested reader may further analyze the assertion that the optimal value of t lies in $\left[0, D\sigma^2\right]$.

To resolve the optimal threshold t for membership inference, we restructure the arguments as follows. Recall from (10) that the adversary aims to ensure both true positive rates and true negative rates are high. We know:

$$\begin{aligned}\mathbb{P}_{X \in \mathcal{D}^+}\left[\mathcal{M}(\mathbf{X}|h(.)) = 0\right] &= \mathbb{P}_{r \sim \mathcal{N}(0, mD\sigma^4)}\left[r < t\right] \\ \mathbb{P}_{X \in \mathcal{S}^+}\left[\mathcal{M}(\mathbf{X}|h(.)) = 1\right] &= \mathbb{P}_{u \sim \mathcal{N}(D\sigma^2, mD\sigma^4)}\left[u > t\right] \\ \mathbb{P}_{(\mathbf{X}, b) \in \mathcal{R}}\left[\mathcal{M}(\mathbf{X}|h(.)) = b\right] &\leq \mathbb{P}\left[u - r > 0\right]\end{aligned} \tag{16}$$

We know that both u, r are sampled from normal distributions. Therefore, define $\gamma = (u - r) \sim \mathcal{N}(D\sigma^2, 2mD\sigma^4)$. This simplifies our discussion to a single normal distribution with mean $\mu_\gamma = D\sigma^2$ and variance, $\sigma_\gamma^2 = 2mD\sigma^4$. We can now calculate the CDF at $x = 0$ to evaluate the maximum probability of success of membership inference.

Let $Z \sim \mathcal{N}(0, 1)$It can hence be shown that

$$\begin{aligned}\mathbb{P}[\gamma > 0] = \mathbb{P}\left(\sigma_t Z + \mu_t\right) &= \mathbb{P}\left(Z > -\frac{\mu_t}{\sigma_t}\right) = 1 - \Phi\left(-\frac{\mu_t}{\sigma_t}\right) \\ &= 1 - \Phi\left(-\sqrt{\frac{D}{2m}}\right)\end{aligned} \tag{17}$$

Clearly, as $m \to \infty$, $\mathbb{P}[\gamma > 0] \to 0.5$. This concludes the proof for Theorem 2.

B.4 Success of Dataset Inference (Theorem 3)

In Theorem 2 we showed that an adversary querying a single data point can say no better than a coin flip about the presence or absence of a given data point in a model's training set. In this section, we show that when we reverse this adversarial game, the victim can utilize the information asymmetry to predict with high confidence if a potential adversary's model stole their knowledge in any form.

First, recall that the victim has access to its own private training set of size m. For the purposes of this proof, we call it $\mathcal{S}_{\mathcal{V}}^m$. As the victim has complete information of the data distribution, it can randomly sample another dataset $\mathcal{S}_0 \subset \mathcal{D} \setminus \mathcal{S}_{\mathcal{V}}^m$.

The victim considers that the potential adversary's model was stolen if the mean 'prediction margin' for the points in $\mathcal{S}_{\mathcal{V}}$ is greater than $\mathcal{S}_0$ by some threshold parameter λ. Let $\psi_{\mathcal{V}}(\mathcal{D}, h(.))$ be $\mathcal{V}$'s decision function.

Recall that, in Theorem 1, we had calculated the expected value of the difference in the prediction margin for the points in the training set versus those in the test set. In the proof of this theorem, we calculate the probability of the mean of the difference being greater than some value λ.

Now, let us calculate the probability of this margin for a data point randomly sampled from the training set. Let $t_{\mathcal{V}}$ represent the mean of the 'prediction margin' of all points in $\mathcal{S}_{\mathcal{V}}$ for a classifier $h(.)$. Similarly, let t_0 represent the mean of the 'prediction margin' of all points in $\mathcal{S}_0$ for the classifier $h(.)$. We will use u_2 to denote the last D dimensions of points in S_0. Then,

$$\begin{aligned} t_{\mathcal{V}} &= \frac{1}{m}\sum_j \left[\left(\sum_i^{i\neq j} y^{(i)}\mathbf{x_2}^{(i)}\cdot\mathbf{x_2}^{(j)}\right) + (\mathbf{x_2}^{(j)})^2\right] \\ &= \frac{1}{m}\sum_j (\mathbf{x_2}^{(j)})^2 + \sum_i \left[\left(\sum_i^{i\neq j} y^{(i)}\mathbf{x_2}^{(i)}\cdot\mathbf{x_2}^{(j)}\right)\right] \\ t_0 &= \frac{1}{m}\sum_j \left[\left(\sum_i^{i\neq j} y^{(i)}\mathbf{x_2}^{(i)}\cdot\mathbf{u_2}^{(j)}\right)\right] \\ \mathbb{P}\left[\psi_{\mathcal{V}}(\mathcal{D}, h(.)) = 1\right] &= \mathbb{P}\left[(t_{\mathcal{V}} - t_2) > \lambda\right] \end{aligned} \tag{18}$$

Recognize the similarity of the above formulation with that discussed in the proof for Theorem 2 in Appendix B.3. Let $t = t_{\mathcal{V}} - t_2$. Then the random variable t represents the a sample from the distribution of means for γ defined in Appendix B.3. We can now directly use the Central Limit Theorem for this proof. Therefore,

$$\begin{aligned} \mu_t &= \mu_z = D\sigma^2 \\ \sigma_t^2 &= \frac{\sigma_z^2}{m} = 2D\sigma^4 \end{aligned} \tag{19}$$

Hence, $t \sim \mathcal{N}(D\sigma^2, 2D\sigma^4)$. It is important to note that this distribution is independent of the number of training points. Hence, unlike membership inference, the success of DI is not curtailed by the lack of overfitting.

Similarly, for an honest adversary, the distribution of 'prediction margin' for points in $\mathcal{S}_{\mathcal{V}}$ is the same as that for the points in $\mathcal{S}_0$. It directly follows that:

$$\mathbb{P}\left[\psi_{\mathcal{V}}(\mathcal{D}, h(.)) = 0\right] = \mathbb{P}\left[\hat{t} < \lambda\right] = \mathbb{P}\left[t > \lambda\right] \tag{20}$$

where, $\hat{t} \sim \mathcal{N}(0, 2D\sigma^4)$. Once again, like the proof of Theorem 2, by symmetry of two normal distributions with the same variance, and shifted means, we can find that the optimal value of the parameter λ that maximized true positives, and minimizes false positives, $\lambda = \frac{\mu_t}{2}$.

Let $Z \sim \mathcal{N}(0, 1)$It can hence be shown that

$$\begin{aligned} \mathbb{P}[\hat{t} > \lambda] &= \mathbb{P}\left(\sigma_t Z + \mu_t > \frac{\mu_t}{2}\right) = \mathbb{P}\left(Z > -\frac{\mu_t}{2\sigma_t}\right) = 1 - \Phi\left(-\frac{\mu_t}{2\sigma_t}\right) \\ &= 1 - \Phi\left(-\frac{\sqrt{D}}{2}\right) \end{aligned} \tag{21}$$

Clearly, as $D \to \infty$, $\mathbb{P}[\hat{t} > \lambda] \to 1.0$. This concludes the proof for Theorem 3.

C MODEL STEALING TECHNIQUES

In this section, we provide more details about the various threat models that we consider in this work. We also provide specific use-cases and motivation for the respective threat models, and introduce a new adaptive adversary targeted specifically against DI.

$\mathcal{V}$: Victim. The victim $\mathcal{V}$ wishes to release its machine learning model to the community, either as a service, or by open-sourcing it for non-commercial academic use. $\mathcal{V}$ wants to ensure that the deployed model is not being misused under the terms of license provided.

$\mathcal{A}_D$: Data Access. The adversary $\mathcal{A}_D$ is able to gain complete access to the victim's private training data, and aims to deploy its own MLaaS by training the same. We note that labeled private training data is one of the most expensive commodities in the deployment cycle of modern machine learning systems.

1. **Model Distillation:** Traditionally, model distillation (Hinton et al., 2015) was used as a method to compress larger models by training smaller students using the logits of a teacher model. We use this as a threat model that the adversary may employ to distance its predictions from a model that was trained using hard labels from the dataset itself. The adversary requires both query access, and access to the victim's private training data for this attack.
2. **Modified Architecture:** Multiple works have attempted at identifying unique properties (or 'fingerprints') of a model by analyzing specific activations and representative features of internal model layers (Olah et al., 2017; 2018; Yin et al., 2019). We study the threat model where the adversary attempts training an alternate architecture on the victim's private dataset $\mathcal{D}_{priv}$ to valid the robustness of our method to changes in model structure.

$\mathcal{A}_M$: Model Access. The use-case of such an adversary is two fold: (1) the victim open-sources their own machine learning model under a license that does not allow other individuals to monetize the same; and (2) the adversary gains insider access to the trained model of a victim. In both the cases, the adversary aims to monetize its own MLaaS and deploys their own model on the web, by modifying the original victim model to reduce the dependence on $\mathcal{K}$.

1. **Fine-tuning:** The adversary has full access to the victim's machine learning model, but not to its training data. While fine-tuning is employed used to transfer the knowledge of large pre-trained models on a given task (Devlin et al., 2018), we use it as a stealing attack, where the adversary uses the predictions of the victim model on unlabeled public data in order to modify its decision boundaries. We consider the setting where the adversary can fine-tune all layers.
2. **Zero-Shot Learning:** This is the strongest adversary that we introduce specifically targeted to evade dataset inference. To the best of our knowledge, we are the first to consider such a threat model. The adversary uses no 'direct' knowledge of the actual training data to avoid any features that it may learn as a result of the training on the victim's private data set. The adversary has complete access to the victim model, and uses data-free knowledge transfer (Micaelli & Storkey, 2019; Fang et al., 2019) to train a student model.

$\mathcal{A}_Q$: Query Access. Model extraction (Tramèr et al., 2016) is the most popular form of model stealing attacks against deployed machine leaning models on the web. We discuss the related work on model extraction attacks in more detail in § A. Depending on the access provided by the machine learning service, an adversary may aim to extract the model using the logits or the labels alone.

1. **Model Extraction Using Labels:** The victim model is used to provide pseudo-labels for a public dataset. The adversary trains their model on this pseudo-dataset. The key difference is that the input data points may be semantically irrelevant with respect to the task labels that the adversary's model is being trained on.
2. **Model Extraction Using Logits:** The performance of model extraction attacks can be improved when the victim provides confidence values for different output classes, rather than the correct labels itself. The adversary's model is trained to minimize the KL divergence with the outputs of the victim on a public (or non-task specific) dataset.

$\mathcal{I}$ **: Independent Model.** Finally, we also study the results of dataset inference on an independent and honest machine learning model that is trained on its own private dataset. This is used as a control to verify that the dataset inference procedure does not always predict that the potential adversary stole the victim's knowledge.[3]

Training the Threat Models. For model extraction and fine-tuning attacks, we use a subset of 500,000 unlabeled TinyImages that are closest to CIFAR10, as created by Carmon et al. (2019). More details about the creation of the dataset can be found in their work. In case of CIFAR100, we use the STL-10 (Coates et al., 2011) dataset to steal the models. We train the student model for 20 epochs in case of model extraction methods and 5 epochs for fine-tuning. For Zero-shot extraction, we use the data-free adversarial distillation method proposed by Fang et al. (2019) and train the student model for 200 epochs. In case of distillation and modified architecture, we have access to the original training data of the victim. We train both models for 100 epochs.

In all the training methods, we use a fixed learning rate strategy. We use the SGD optimizer and decay the learning rate by a factor of 0.2 at the end of the $0.3 * n$, $0.6 * n$ and $0.8 * n$ epochs, where n is the total number of epochs that the model is trained for.

D EMBEDDING GENERATION

Embedding Generation Hyperparameters.

For the case of ***MinGD*** attack, we perform adversarial attacks defined by the optimization equation

$$\min_{\delta} d(x, x+\delta) \;\; s.t. \;\; f(x+\delta) = t; \;\; x+\delta \in [0,1]^n \tag{22}$$

The distance metric $d(x, y)$ refers to the ℓ_p distance between points x and y for $p \in \{1, 2, \infty\}$, and t is the targeted label. To perform the optimization, we perform gradient descent with steps of size α_p. We take a maximum of 500 steps of gradient optimization, but pre-terminate at the earliest misclassification. The step sizes for the individual perturbation types are given by $\{\alpha_\infty, \alpha_2, \alpha_1\} = \{0.001, 0.01, 0.1\}$.

For the case of ***Blind Walk***, We sample a random initial direction δ. Starting from an input (x, y), take k steps in the same direction until $f(x+\delta) = t; \; t \neq y$. Then, $d(x, x+k\delta)$ is used as a proxy for the 'prediction margin' of the model. We repeat the search over multiple random initial directions to increase the information about a training data point's robustness, and use each of these distance values as features in the generated embedding.

As an implementation detail, we sample between uniform, laplacian and gaussian noise to generate embedding features. To measure the final perturbation distance from the initial starting point, we use different ℓ_p norms for each of the noise sampling methods. For uniform noise, we compute the ℓ_∞ distance; for gaussian noise, the ℓ_2 distance; and for laplacian noise, the ℓ_1 distance of the nearest misclassification. We take k steps of *Blind Walk* up till misclassification. However, we do not exceed more than 50 steps and prematurely terminate without misclassification in the event that the prediction label does not change.

Performance of White Box Approach. We find in our evaluations that the white-box **MinGD** method generally underperforms the *Blind Walk* method. This happens despite its ability of being able to compute the nearest distance to any target class more accurately. While on the onset, this may seem to be a counter-intuitive result, since generally with more access, the performance of mapping the neighbours should only increase.

However, we note an important distinction. The end goal of the query generation process is not to calculate the minimum distance to target classes accurately, but rather to understand the 'prediction margin' or the local landscape of a given data point. Readers may recall from adversarial examples literature (Szegedy et al., 2013) that adversarial examples can easily be constructed on the dataset that a given machine learning model was trained on. This observation hurts the idea of Figure 1b.

[3]Note that since we consider the difference in the distribution of outputs of the auxiliary classifier on embeddings from the test and training set (rather than hard labels from the auxiliary classifier), even in the absence of this control, we can un-deniably verify the confidence of dataset inference. This is only included to contrast the difference and make the effects of the method clearer to the reader.

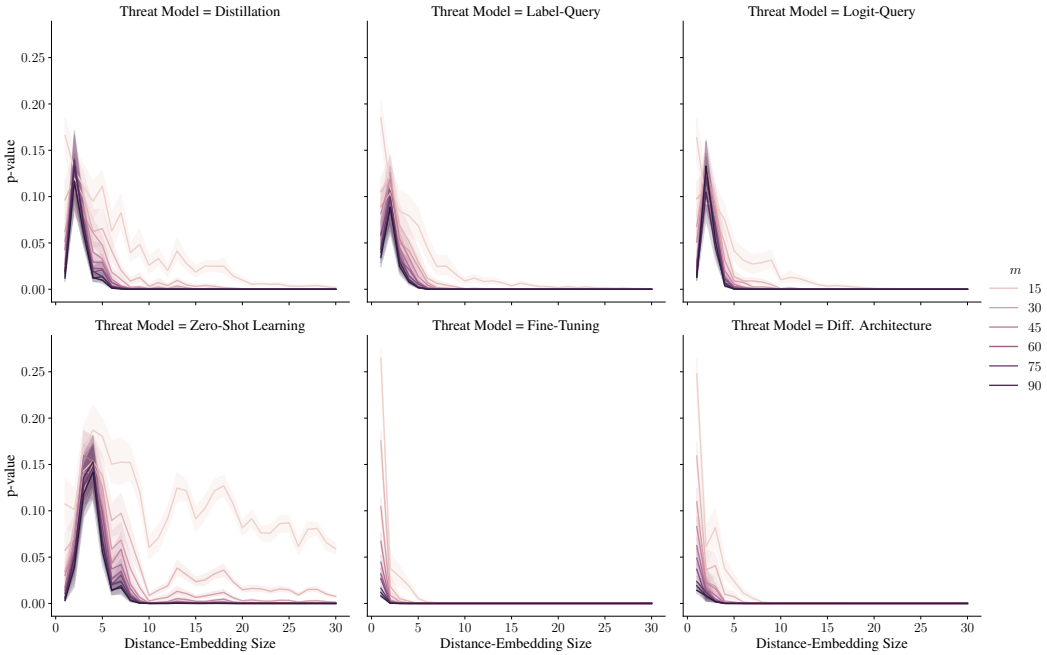


Figure 4: p-value vs. distance-embedding size

Despite pushing the neighbouring class boundaries away, the existence of adversarial examples elucidates the existence of small pits within the landscape of the model.

We hypothesize that the gradient-based optimization procedure, finds these adversarial regions that are not 'truly' representative of the neighbouring region or the 'prediction margin' of a given data point. We hypothesize that on the contrary *Blind Walk* is able to perform a spectacular job at the same end goal. Since we are no longer adversarially trying to optimize the minimum distance to the neighbouring classes, multiple *Blind Walk* runs effectively map the **'average case'** prediction margin, which we argue is more useful than the **'worst case'** prediction margin as obtained by **MinGD**.

E Additional Results

E.1 Effect of Embedding Size

For all models, richer embeddings reduce the need for more reveled samples. (See Figure 4). We note that in the main body of this work, we had used a fixed size of embedding vector, with 30 input features. However, recall that in the black-box setting, the victim incurs additional cost for querying the potential adversary. Therefore, in this section we aim to understand the marginal utility of extra embedding features added. In general, we find that for most of the threat models studied, using only 10 features for the embedding space is sufficient to achieve the required threshold p-value of 0.01. This suggests that we can slash the number of queries made to the potential adversary by one-thirds, without loss in confidence of prediction.

Interestingly, we also note that even in scenarios where the victim reveals only 15 samples, additional embedding features have insignificant advantage as opposed to querying fresh samples. This suggests that the amount of entropy gained by revealing a new data point is significantly more than that by extracting more features (beyond 10) for the same data point. We also note that the effect is not-consistent in the Zero-shot Learning threat model.