SEAT: Similarity Encoder by Adversarial Training for Detecting Model Extraction Attack Queries

Zhanyuan Zhang
UC Berkeley
zhang_zhanyuan@berkeley.edu

Yizheng Chen
UC Berkeley
1z@berkeley.edu

David Wagner
UC Berkeley
daw@cs.berkeley.edu

ABSTRACT

Given black-box access to the prediction API, model extraction attacks can steal the functionality of models deployed in the cloud. In this paper, we introduce the SEAT detector, which detects black-box model extraction attacks so that the defender can terminate malicious accounts. SEAT has a similarity encoder trained by adversarial training. Using the similarity encoder, SEAT detects accounts that make queries that indicate a model extraction attack in progress and cancels these accounts. We evaluate our defense against existing model extraction attacks and against new adaptive attacks introduced in this paper. Our results show that even against adaptive attackers, SEAT increases the cost of model extraction attacks by 3.8 times to 16 times.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Artificial intelligence**;

KEYWORDS

Model Extraction; Black-box Attacks; Adversarial Machine Learning; MLaaS; Intellectual Property

ACM Reference Format:
Zhanyuan Zhang, Yizheng Chen, and David Wagner. 2021. SEAT: Similarity Encoder by Adversarial Training for Detecting Model Extraction Attack Queries. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security (AISec '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3474369.3486863

1 INTRODUCTION

Deep Neural Networks (DNNs) have seen tremendous success in many applications such as image recognition, language translation, and voice assistants. Many cloud platforms (e.g., Microsoft Azure, Google Cloud Platform, Amazon Web Services, Clarifai) provide Machine Learning as a Service to make these models more accessible to users: they train a model, and allow their customers to submit an image and obtain the model's classification of that image. Researchers have demonstrated that model extraction attacks allow malicious customers use this service to learn a model with similar performance as the cloud provider's model, effectively stealing the valuable intellectual property of the model owner. These attacks need only the ability to query the model (i.e., submit images and obtain their classification), without requiring knowledge of the full training set, model architecture, or other hyperparameters.

We study how to detect model extraction attacks. We especially focus on Jacobian-based Augmentation (JBA) attacks [22, 37, 50], one important approach for model extraction. In a JBA attack, the attacker uses data augmentation to iteratively expand a set of seed samples into a substitute training set. In each iteration of data augmentation, the attacker constructs new data samples near the model's decision boundary by augmenting existing samples, queries the model in the cloud to label the augmented samples, and then trains a substitute model on these data points. Research shows that JBA attacks can be effective, even when the attacker has very little training data of their own.

In this paper, we propose SEAT (Similarity Encoder by Adversarial Training), an algorithm to detect the queries generated by model extraction attacks. Our intuition is, since JBA attackers augment existing samples to synthesize new samples, they inevitably generate similar pairs of samples. Therefore, to detect attacks in progress, we compare the similarity of each incoming query to historical queries made by the same account. Specifically, we train a similarity encoder that can detect similar pairs of samples. Then, our detector produces an alert if an account has issued more than K pairs of similar queries. We assume that the cloud provider will terminate any account that has been detected in this way as involved in malicious activity. Since it is not trivial for the adversary to create a new account, we can increase the cost of model extraction attacks by requiring them to obtain many accounts in order to extract a highly accurate model. We show that we can achieve a false positive rate of 0.01% for benign queries, ensuring we do not harm the utility of the protected model for benign users.

We evaluate the effectiveness of SEAT against state-of-the-art model extraction attacks. We focus on settings where the attacker has access to only a small number of labelled training samples (e.g., 1% of the size of the victim's training set). In this setting, we thoroughly evaluate the effectiveness of the SEAT detector against multiple JBA attacks. Our experiments demonstrate that SEAT detector effectively detects malicious accounts sending JBA-based model extraction queries. Furthermore, we evaluate our method against data-free model extraction attack [45] and show that SEAT detector is also able to detect these attacks as well.

We also examine whether adaptive attacks can defeat SEAT. We consider both query blinding (QB) attacks, previously introduced by Chen et al. [7], and a novel attack we introduce called query filtering (QF). In a query filtering attack, the attacker tries to predict which queries will trigger the SEAT defense and sends only queries that are predicted to not raise any alarms. Specifically, the attacker trains their own substitute similarity encoder using training data available

AISec '21, November 15, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8657-9/21/11.
https://doi.org/10.1145/3474369.3486863

to them and emulates the SEAT detector on each query before submitting it to the cloud. The intuition is that if the attacker's substitute encoder is similar enough to the defender's encoder, then this may help the attacker evade detection. Our experiments show that these adaptive attacks are not able to defeat SEAT. Even when using the adaptive QF attack, the attacker would need 33 fake accounts to successfully extract the model when it is defended by SEAT, compared to only one account for an undefended model. QB attacks are somewhat more successful, but an attacker using QB would still need 23 fake accounts to successfully extract the model.

We show that SEAT is more effective than other existing detectors. We compare against state-of-the-art model extraction attack detectors that can operate in an account-based setting, specifically, the out-of-distribution detector (OOD) [1] and PRADA [22]. Our experimental results show that SEAT increases the fake accounts needed by adaptive attackers to successfully extract a model by 3.8–16×, compared to state-of-the-art detectors.

In summary, we make the following contributions:

- We propose SEAT, a new method to detect black-box model extraction attacks. We show it can detect Jacobian-based augmentation attacks and data-free model extraction attacks.
- We show that Jacobian-based augmentation attacks are most effective when the attacker has access to only a very small set of samples from the original training set.
- We propose a new adaptive attack, the query filtering attack, that is specifically designed to evade the SEAT detector. Our results show that the SEAT detector is still effective against this adaptive attack.

In the rest of the paper, we introduce the threat model and related works (section §2), explain our SEAT detector (section §3), evaluate its security against adaptive attacks, and finally presents experimental results demonstrating that SEAT is both effective and robust (section §4).

2 BACKGROUND & RELATED WORKS

In this section, we give an overview on model extraction attack algorithms and existing defenses from prior works.

2.1 Notations

In this paper, we denote the victim's model as F_V, the extracted model as F_A, the victim's training set as X_V.

2.2 Threat Model

2.2.1 Adversary's Goal. The adversary aims to obtain an extracted model that performs well on the victim's task. Specifically, the adversary wants to maximize the accuracy of the extracted model on the victim's test set, which we denote as the *extraction accuracy*. Other goals [21], such as obtaining high prediction agreement between the victim model and the extracted model [8, 21, 22, 37, 44] or exactly recovering the victim's weights [2, 4, 28, 31, 44], are not the focus of this paper.

2.2.2 Account-Based Defense. We focus on the account-oriented setting. We assume a user must create an account to query the prediction API. The account could be associated with a credit card, a valid phone number, or a mobile device ID, so if we detect misbehavior, we can block the user's account and it is not trivial to create a new account. An attacker could create multiple fake accounts [7, 11, 27, 49], and with enough fake accounts any defense could be circumvented, but we assume the cost of doing so scales with the number of fake accounts. Every time our system detects that an account is behaving maliciously, the account making the queries is blocked and the attacker must create a new account to continue the attack. Thus, we measure the effectiveness of a defense scheme by the number of fake accounts an attacker would need to successfully extract an accurate model. We aim to increase this number to design a successful detection system, even when attackers have knowledge of the defense and perform adaptive attacks.

2.2.3 Seed Samples. We assume the attacker has access to a limited number of data samples. We call these the seed samples. These may be a subset of the training set used by the cloud provider or samples from the same distribution; or they may be from some other publicly available dataset with a different distribution. We assume the number of seed samples is strictly limited: e.g., from 1% to 10% of the size of the victim's training set.

2.2.4 Black-box Hard-label Attack. We assume the adversary has black-box query access to the victim's model via the prediction API. We assume the API only provides *hard label* response, i.e., the API returns the predicted label without prediction probabilities. Removing prediction probabilities makes attacks harder, thereby requiring more queries for successful model extraction, or reduces the extracted model's accuracy [44].

As an exception, when we evaluate SEAT against data-free attacks, we give the attacker more advantage by providing the prediction probabilities (section §2.4), demonstrating that SEAT can detect data-free attacks even when prediction probabilities are returned.

2.2.5 Query Budget. We assume that there is a limit to how many queries an attacker can make to the prediction API, which we denote as the query budget B.

2.2.6 Query-based Attacks. We focus on detecting model extraction attacks that work by making queries to the prediction API. Other attacks, such as functionally-equivalent extraction attacks [4, 21] and hardware side channel attacks [48], are out of the scope of this paper. There are three main categories of query-based attacks, which we discuss next.

2.3 Jacobian-based Augmentation Attacks

Many existing model extraction attacks fall into the category of Jacobian-based Augmentation (JBA) attack [22, 37, 50].

2.3.1 Attack Algorithm. The JBA attacker constructs a surrogate training set by iteratively expanding the seed query set based on information returned from the victim model's prediction API.

Algorithm 2 presents the details of JBA attack. The algorithm takes what the attacker has access to as input: a seed query set X_0, prediction API of the victim's model F_V, and the query budget B. In addition, the attack also requires an augmentation algorithm Aug, and parameters to indicate the number of training epochs within

Algorithm 1 Augmentation Algorithm

Input: Original data x;
Extracted F_A;
Loss function $\mathcal{L}$;
Number of Steps k;
Step Size α
Output: Synthesized data $x\prime$
for step in $1, 2, \ldots, k$ **do**
  $x' \leftarrow x' - \alpha \cdot \text{sign}(\nabla_x(\mathcal{L}(x, F_A(x))))$
end for
return x'

each augmentation round (n) and after all rounds (N). As output, Algorithm 2 returns the extracted model F_A.

First, the attacker queries the victim model to obtain the labels for the seed query set, and initializes the surrogate training set D_T with labeled seed samples (step 1 and step 2). This uses m queries out of the total query budget B. Then, the attacker initializes and trains the extracted model for n epochs (step 3). From step 4 to step 11, the attacker iteratively uses an augmentation algorithm to generate new samples, queries the new samples, expands the surrogate training set, and retrains the extracted model F_A for n epochs, until the query budget is exhausted. Finally, the attacker obtains a surrogate model by reinitializing F_A and training it on the full augmented query set (step 11).

The augmentation algorithm is essential to the JBA attack (step 7, Algorithm 2), since the adversary relies on augmented data to explore some important properties (*i.e.* decision boundary) of the victim's model. The augmentation algorithm takes the existing surrogate training samples D_T and the current version of the extracted model F_A, and outputs the set of augmented samples. Since the adversary has only black-box access, the current F_A serves as a proxy to approximate the gradients of the victim's model. Algorithm 1 shows the procedure to perturb each data point x within D_T to generate a new augmented sample x', via the guidance of the Jacobian calculated from some loss function $\mathcal{L}(x, F_A(x))$.

2.3.2 Attack Variants. Different Jacobian-based extraction algorithms mainly differ in their loss functions $\mathcal{L}$, which are usually designed based on the loss functions for crafting adversarial examples.

Papernot *et al.* [37] and Juuti *et al.* [22] proposed loss functions similar to those used by targeted and non-targeted Projected Gradient Descent attack (PGD)[29]. Yu *et al.* [50] investigated using loss functions from the Carlini-Wagner ℓ_2 attack (CW_ℓ_2) [5] and the Feature Adversary Attack[42]. Yu *et al.* [50] also proposed Feature Fool (FF), which constructs the loss function as the ℓ_2 distance between the original data x and augmented data x' plus a triplet margin loss [46].

2.4 Data-free Model Extraction Attacks

Data-free model extraction (DFME) attacks construct a surrogate training set by synthesizing samples and querying the victim for the labels of these samples, without requiring access to a seed query set. For example, the adversary can synthesize samples using line search, or by training some generative models.

Algorithm 2 Jacobian-based Augmentation Attacks

Input: Seed query set $X_0 = \{x_1, x_2, \ldots, x_m\}$;
Prediction API of the victim's model F_V;
Query budget B;
Augmentation Algorithm Aug;
n training epochs for each augmentation round;
N training epochs after all rounds
Output: Extracted Model F_A
1: $Y_0 \leftarrow F_V(X_0)$ (Send queries to F_V)
2: $D_T \leftarrow \{(X_0, Y_0)\}$
3: Initialize F_A and train it on D_T for n epochs
4: **while** Number of queries < B - m **do**
5: Initialize X_Q as a set
6: **for** for all x in D_T **do**
7: $x_q \leftarrow Aug(x, F_A; \mathcal{L}, k, \alpha)$ (Generate new query)
8: $y_q \leftarrow F_V(x_q)$ (Send queries to F_V)
9: $D_T \leftarrow D_T \cup \{(x_q, y_q)\}$
10: **end for**
11: Reinitialize F_A and train it on D_T for n epochs
12: **end while**
13: Reinitialize F_A and train it on D_T for N epochs
14: **return** F_A

Tramèr *et al.* [44] proposed an algorithm to extract shallow neural networks. They first generate some random data points, and then synthesize additional data using line search among these random points. In their implementation[1], line search is essentially binary search to find the middle point between two samples.

Truong *et al.* [45] proposed a data-free model extraction attack based on Generative Adversarial Networks (GANs) [13]. The adversary trains a generator G to generate samples that maximize the disagreement between the victim's model F_V and the extracted model F_A, and also trains F_A to minimize such disagreement. The disagreement is captured by a loss function $\mathcal{L}$. The adversary's objective function is

$$\min_{F_A} \max_{G} \mathbb{E}_{z \sim \mathcal{N}(0,1)} [\mathcal{L}(F_V(G(z)), F_A(G(z)))] \tag{1}$$

The adversary alternates between training G and training F_A, until the query budget is exhausted. Note that equation (1) involves F_V, but the adversary only has black-box access to the victim's model, so they need extra queries to approximate the gradient $\nabla_x F_V(x)$ by the Forward Differences method [12]. Specifically, they compute the gradient by independently sampling m directions Z_i from the standard normal distribution:

$$\nabla_{\text{FWD}} F_V(x) = \frac{1}{m} \sum_{i=1}^{m} \frac{F_V(x + \epsilon \cdot Z_i) - F_V(x)}{\epsilon} Z_i \tag{2}$$

2.5 Sampling-based Extraction Attacks

Researchers have also proposed model extraction attacks that use specialized sampling techniques. Unlike JBA and DFME attacks, sampling-based attacks do not generate data, but rather, they learn some *sampling strategy* to sample *informative* data from their query sets, and construct a surrogate dataset by using the victim model to label the data. The adversary then trains the extracted model on

[1] https://github.com/ftramer/Steal-ML/blob/master/neural-nets/utils.py#L296

the surrogate dataset. Researchers have proposed several ways to learn the sampling strategy.

Correia-Silva *et al.* [8] proposed *CopyCat*, which randomly samples from datasets that may or may not be in the same problem domain as the victim, and queries the victim for these samples. People often call this strategy the *random sampling strategy*, and it usually serves as a baseline to compare against for other strategies.

Orekondy *et al.* [34] proposed *KnockoffNet*, which uses an adaptive strategy to select samples to query. The reward function for learning the strategy encourages selecting high-confidence samples, diverse samples, and samples that reveal the difference between the victim model and the extracted model.

Pal *et al.* [36] proposed the Uncertainty, K-center, DFAL, and DFAL+K-center strategies in their *ActiveThief*. These strategies evaluate all the samples in the adversary's query set and collect the best k samples by their evaluation functions.

Detecting sampling-based attacks is not the focus of this paper, since sampling-based attacks only query natural images and require many seed images and thus fall outside our threat model.

2.6 Detection Schemes

In this section, we introduce existing defenses or detection schemes against model extraction attacks.

2.6.1 PRADA. Juuti *et al.*[22]'s PRADA detects model extraction attacks based on the observation that the ℓ_2 distance among benign queries follows normal distribution. PRADA monitors the distribution of $\min_{x \in \text{history}} \|x - x_{\text{new query}}\|_2$ and checks the normality of this distribution with the Shapiro-Wilk normality test. Unfortunately, Chen *et al.* [7, § 5.3] show that an attacker using a query blinding attack can avoid being detected by PRADA, and Yu *et al.* [50, § VI.A] show that the attacker can bypass PRADA by manipulating the query distribution.

2.6.2 Out-of-distribution Detector. Atli *et al.* [1] and Kariyappa *et al.* [24] proposed out-of-distribution (OOD) detectors. Atli *et al.* train a binary classifier to distinguish whether a query comes from the same distribution as the victim's training set. Kariyappa *et al.* fine-tune the protected model such that it tends to output high top-1 prediction confidence on *in-distribution* data, whereas the top-1 prediction confidence for OOD data is uniformly distributed from 0 to 1. While Atli *et al.* do not propose what the defender should do when OOD queries are detected, in *Adaptive Misinformation* [24], the defender returns inconsistent labels to those detected OOD queries. Essentially, Adaptive Misinformation assumes all OOD queries are adversarial. In comparison, we assume all the natural data are benign in this paper.

2.7 Other Defenses

2.7.1 Prediction Poisoning/Modification. Orekondy *et al.* [35] propose prediction poisoning, which perturbs the prediction probabilities returned by the victim model. Specifically, given a loss function L, they add perturbation ϵ^* to a output y while maintaining API utility, where ϵ^* is:

$$\epsilon^* = \max_{\epsilon} \angle(\nabla_{F_A} L(F_A, y + \epsilon), \nabla_{F_A} L(F_A, y))$$

In other words, ϵ^* represents misinformation that gives a wrong direction to adversaries when they use this contaminated output to train F_A on some loss L. One critical weakness is that an attacker can easily post-process the output into a one-hot vector, and ignore the probabilities, and mount any hard-label model extraction attack.

Kariyappa *et al.* [23] defend against model extraction attacks by detecting queries that do not come from the same distribution as the training data and treating them as malicious. They train an Ensemble of Diverse Models (EDM) to produce accurate predictions on in-distribution data, but yield inconsistent prediction on out-of-distribution (OOD) data. Each query is assigned to one of these models based on a secret hash.

2.7.2 Detectors against Black-box Attacks. Black-box adversarial example attacks [3, 6, 19, 32] send a sequence of queries to the victim model in order to craft adversarial examples. Our similarity-based detector is inspired by Chen *et al.*'s Stateful Detector (SD) for detecting adversarial example attacks [7]. SD maps each query to a feature vector using a similarity encoder and stores the vectors for all past queries. For each query, it computes the average distance to its k-nearest neighbors; if this distance is below some threshold, it treats the query as malicious and cancels the account that issued the query. In Section §4.8, we show that SD is not effective at detecting model extraction attack queries, and our SEAT detector is more suitable for this task.

Blacklight by Li *et al.* [27] is another method for detecting black-box adversarial example attacks. Blacklight also measures the similarity among adversarial queries, measuring similarity using the L_2 distance rather than using perceptual similarity measures. We do not use L_2 distance in our defense because it can be evaded with query blinding attacks.

3 METHODOLOGY

In this section, we introduce the SEAT (Similarity Encoder trained by Adversarial Training) detector to defend against black-box model extraction attacks. We monitor incoming queries and terminate the user's account if any suspicious behavior is detected from the account.

The SEAT detector has two components: 1) a similarity encoder helps detect similar queries (Section §3.1) and 2) a detection scheme monitors the number of similar pairs (Section §3.2).

3.1 SEAT

We train a similarity encoder to detect similar images, such as might be generated by a model extraction attack. We observe that JBA attacks augment queries in ways that are similar to crafting adversarial examples. Motivated by this observation, we train a similarity encoder to recognize these queries, using adversarial training.

To train our similarity encoder, we start from taking the victim model without the last classification layer as a pre-trained similarity encoder, and then fine-tune the model using adversarial training. During fine-tuning, we minimize a contrastive loss that encourages a small distance between the embeddings of similar pairs of samples, and a large distance between embeddings of different pairs. We construct similar pairs of samples using the projected gradient descent (PGD) attack [29] that generates adversarial examples.

Our contrastive loss function follows previous works that train a similarity encoder to recognize visually similar images [7, 15]:

$$\mathcal{L}(x_0, x_+, x_-, f) = \|f(x_0) - f(x_+)\|_2^2 + \max\{0, m^2 - \|f(x_0) - f(x_-)\|_2^2\} \quad (3)$$

In equation (3), m is a constant, f is a similarity encoder, x_0 is a natural image, x_+ is a positive sample which f should consider closed to x_0, and x_- is a negative sample, a different natural image from x_0 that f should consider far away from x_0. While Chen *et al.* [7] show that generating x_+ by image random transformations (*e.g.* rotation, scaling, cropping, etc.) is sufficient for detecting black-box adversarial examples, we empirically find that this is insufficient for detecting model extraction attacks (more details in section §4.5). Instead, we generate x_+ by projected gradient descent (PGD) attack and show that using adversarial training significantly improves the effectiveness of the similarity encoder to detect model extraction queries. We find that setting $m = \sqrt{10}$ is sufficient.

3.2 Similar Pairs

The second component of our detector is a detection scheme based on similar pairs. JBA model extraction attacks generate many pairs of similar images: they take each seed image x_i, and augment it to a perturbed image x_i', so during a JBA attack we expect to see many pairs (x_i, x_i') of similar queries to the victim model. Therefore, we count the number of similar pairs of queries.

We use the similarity encoder to encode all queries from an account and log them all. For each new query, the defender checks whether this new query is similar to any historical query from the same account. A similar pair (q_i, q_j) is a pair of queries where the ℓ_2 distance between their feature vectors $f(q_i)$ and $f(q_i)$ is less than some threshold δ, i.e., $\ell_2(f(q_i), f(q_j)) < \delta$. Once the number of similar pairs exceeds some threshold of tolerance N_{thresh}, our detector will raise an alarm and we will cancel that account. The defender can tune N_{thresh} and δ to control the false positive rate (FPR). We choose to keep N_{thresh} fixed to 50 and tune δ. The smaller δ is, the more sensitive the detector will be, and thus the higher its FPR. We use binary search to find the best δ such that FPR $\approx 0.01\%$ for benign queries.

Note that our strategy differs from that used by SD [7] for detecting adversarial examples. Attacks for generating adversarial examples typically produce one large cluster where all queries are similar to each other, so SD raises an alarm if the k-nearest neighbor distance of a query is below some threshold. In contrast, model extraction attacks produce a large number of small clusters, one cluster per seed image (with each cluster containing possibly as few as 2 images). This makes counting the number of similar pairs more appropriate for detecting model extraction attacks. See Figure 1 for a visual comparison.

3.3 Adaptive Attacks

An effective detector should be robust enough to detect adaptive attacks, where the attacker is aware about the defense technique and employs adaptive strategies to evade the detection. In this section, we first introduce *query filtering*, a new adaptive attack designed to target the SEAT detector (section §3.3.1). Then, we summarize another adaptive attack proposed by Chen *et al.* [7], *query blinding* (section §3.3.2).

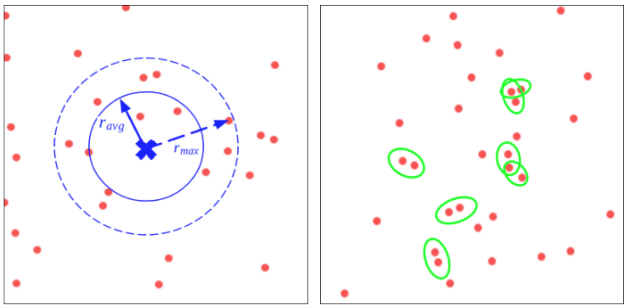

Figure 1: Illustration of detection schemes. Left: SD [7] measures r_{avg}, the average ℓ_2 distance to the k nearest queries, and cancels the account if it is below some threshold. Right: SEAT counts the number of similar pairs (circled in green), i.e., pairs whose ℓ_2 distance is less than some threshold. Since similar pairs of queries in model extraction attacks do not necessarily form a single cluster, SEAT is better suited for detecting model extraction attacks.

Both of these adaptive attacks can be used by any model extraction attack. The attacker can use query filtering before sending the queries to the victim's model F_V (e.g., step 7 in Algorithm 2) to eliminate or modify queries that may be caught by the SEAT detector.

3.3.1 Query Filtering. We propose *query filtering* (QF), a new adaptive attack strategy designed to evade the SEAT detector. In this attack, the attacker locally emulates the detector and removes those queries that are predicted to raise the detector's alarm. For instance, if the attacker knows that the defender is using a similarity encoder to detect similar queries, the attacker can build a substitute detector D_{sub} by training a substitute similarity encoder E_{sub} and tuning a substitute detection threshold δ_{sub}. Then, the attacker can remove those queries which D_{sub} believes are suspicious. Depending on the threat model, the attacker can train this D_{sub} on either a subset of F_V's training set X_V, or a publicly accessible dataset that is descriptively similar to X_V. We believe QF is a very strong adaptive attack since it allows the attacker to take advantage of full knowledge of how the victim sets up the defense.

3.3.2 Query Blinding. Query blinding (QB), proposed by Chen *et al.* [7], is another adaptive attack strategy against similarity-based detectors. QB hides an actual query x by apply some random transformation function t. In particular, the adversary sends $x' = t(x; r)$ to F_V, where r is a parameter that controls the amount of distortion introduced. The transformation is designed so that $F_V(x')$ will hopefully be similar to $F_V(x)$, yet x' will be very different from x. With QB, step 7 in Algorithm 2 becomes $\mathcal{Y}_T \leftarrow F_V(t(\mathcal{X}_T; r))$ and the rest of the algorithm remains unchanged.

Following [7], we choose a wide range of random image transformations (*e.g.* rotation, scaling, random noise, etc.) and apply their default r values for the query blinding attacks. We also follow the setting and architecture in [7] to examine an auto-encoder blinding function. We train the auto-encoder blinding function t_{auto} such that for a model F we have $F(t_{\text{auto}}(x, r)) \approx F(x)$. As illustrated

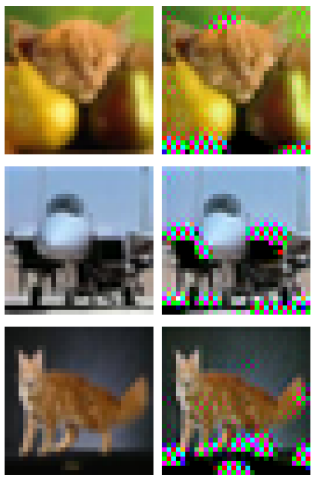

Figure 2: Left: Original input images; Right: Images after query blinding is applied. We use an auto-encoder that learns to allocate random noise to positions in the input that have minimal impact on the model output.

in figure 2, a well-trained auto-encoder blinding function can redistribute the input noise so that it has minimal impact on model outputs. An adaptive attacker can train an auto-encoder t_{auto} and substitute model F using either a subset of F_V's training set X_V, or a publicly available dataset that is closed to X_V, depending on the threat model.

4 EXPERIMENTS

In this section, we evaluate the SEAT detector from the following aspects. We will first study the effect of different threat models where the attacker has different levels of access to seed samples (Section §4.2). Then, we describe how we train the similarity encoder to make SEAT as effective as possible (Section §4.5), and evaluate its false positive rate (Section §4.4). We show that the SEAT detector can effectively detect non-adaptive attacks (Section §4.5), as well as adaptive query filtering and adaptive query blinding attacks (Section §4.6). Moreover, SEAT detector can also detect data-free model extraction attacks (Section §4.7). Lastly in Section §4.8, we compare the SEAT detector to other baseline defenses, and in Section §4.9, we conduct a ablation study to show that the pair-based detection scheme in SEAT detector outperforms a cluster-based one as used in SD.

4.1 Experiment Settings

4.1.1 Model Architectures. In all experiments for random sampling (RS) and JBA attacks, the victim model is VGG16 [43] with batch normalization [20]. We use a pre-trained CIFAR10 classifier as victim model[2], which has 93.38% accuracy on the test set. For simplicity, we denote it as VGG16 in the rest of the paper. We allow the attacker to use the same model architecture as the victim for the extracted model, and to start training from weights pre-trained on ImageNet [10].

[2]pre-trained weights can be downloaded here: https://github.com/tribhuvanesh/prediction-poisoning#victim-models

Similarity encoder in SEAT. We train a similarity encoder by starting from the victim's model (*i.e.* same architecture and weights) without the last classification layer, and then fine-tune the encoder by adversarial training, as described in section §3.

Surrogate similarity encoder. For our experiments with Query Filtering (section §3.3.1), the attacker trains a surrogate similarity encoder. We show the experiment results by two architectures for the surrogate encoder: VGG16 without the last layer (the same architecture as the victim's encoder), and the 5-layer CNN architecture applied in [7] (which we denote as *5-layer*).

4.1.2 Model Extraction Attack Algorithms. We run three types of model extraction attack algorithms: random sampling (RS), different variants of the Jacobian-based Augmentation (JBA) attack, and data-free model extraction (DFME) attack.

Random Sampling (RS): The RS attack randomly selects some samples from a dataset that the attacker has access to, and queries the victim model to label the samples (Section §2.5). Using the labeled samples, the attacker trains an extracted model. In our experiments the attacker has either 500 or 5,000 samples. We experiment with 9 different datasets as described in Section §4.2.

JBA: As discussed in section §2.3, the JBA attack can use any algorithm to craft adversarial examples. We experiment with three different variants„ JB-top3, JBA-PGD and JBA-CW_l2, named according to the technique used to generate adversarial examples. In each iteration of the attack, we choose the top-3 predictions of the extracted model rather than the victim's to craft targeted adversarial examples using some augmentation algorithms.

JB-top3: Following [35], we rename *I-FGSM* [22] to *JB-top3*. We adapt the implementation and hyperparameters from Orekondy *et al.* [35].

JBA-PGD and **JBA-CW_l2:** We use PGD [29] and CW ℓ_2 attacks, as implemented by Foolbox3 [40, 41] in PyTorch [38]. Unless specified otherwise, we use the default settings in Foolbox3 PyTorch. In JBA-PGD we use targeted ℓ_∞ PGD with default settings and $\epsilon = 8/255$ (unless specified otherwise). In JBA-CW_l2, we use targeted CW ℓ_2 attack [5], with 150 steps and 5 binary search steps.

Data-free model extraction (DFME): We use the official implementation of DFME[3]. Both the victim and adversary use Resnet34-8x [16]; the victim model achieves 95.54% accuracy on the CIFAR10 test set[4].

All the accuracy scores (in %) are rounded to their nearest integers, and vary within ±1%.

4.2 Seed Query Set

We experiment with 9 different seed query sets to simulate different dataset access an attacker might have, listed in the first column of Table 1. They include the subset of victim's training set CIFAR10, and 8 other publicly available datasets of different complexity ([9, 10, 14, 25, 26, 33, 39, 47]). Some datasets are descriptively similar to the victim's training set, such as CINIC10; whereas, other datasets are very different from the victim's training set, e.g., SVHN, Indoor67, and CUBS200. We reshape images to 32×32 to be consistent with inputs to the victim's model trained on CIFAR10. We compare

[3]https://github.com/cake-lab/datafree-model-extraction

[4]The pre-trained weights of the victim can be found at https://github.com/VainF/Data-Free-Adversarial-Distillation#0-download-pretrained-models-optional

Table 1: Accuracy of CIFAR10 models extracted by RS (random sampling baseline), JB-top3, JBA-PGD attackers, when they have access to different seed query sets. We compare 500 seed samples vs 5,000 seed samples randomly selected from 9 different datasets. Given access to 500 samples from the original CIFAR10 training set, the JBA-PGD attacker improves the extraction accuracy by 7% compared to the RS baseline, significantly higher than 2% gain in the 5,000 seed samples setting.

Number of Seed Samples		500					5,000					50,000†
Seed Query Set	**Size**	**RS**	**JB-top3**		**JBA-PGD***		**RS**	**JB-top3**		**JBA-PGD***		**RS**
		Acc	Acc	Gain	Acc	Gain	Acc	Acc	Gain	Acc	Gain	Acc
CIFAR10 [25]	50,000	66%	69%	+3%	73%	**+7%**	85%	87%	**+2%**	87%	**+2%**	N/A
CINIC10[9]	70,000	58%	64%	+6%	69%	**+11%**	78%	84%	**+6%**	84%	**+6%**	89%
TinyImageNet[26]	100,000	46%	57%	+11%	61%	**+15%**	69%	78%	**+9%**	78%	**+9%**	84%
ImageNet1k[10]	1,281,167	44%	57%	+13%	62%	**+18%**	67%	78%	**+11%**	78%	**+11%**	85%
CIFAR100 [25]	50,000	43%	55%	+12%	58%	**+15%**	68%	78%	**+10%**	78%	**+10%**	84%
Caltech256[14]	30,607	44%	49%	+5%	58%	**+14%**	63%	77%	**+14%**	77%	**+14%**	78%
Indoor67[39]	14,280	33%	44%	+11%	51%	**+18%**	50%	70%	+20%	71%	**+21%**	63%
CUBS200[47]	6,033	20%	23%	+3%	31%	**+11%**	31%	46%	+15%	54%	**+23%**	36%
SVHN[33]	23,380	19%	24%	+5%	32%	**+13%**	24%	40%	+16%	43%	**+19%**	36%
Number of Queries		500	50,000		50,000		5,000	50,000		50,000		50,000†

* JBA-PGD uses $\epsilon = 8/255$. † Caltech256, Indoor67, CUBS200, and SVHN have fewer than 50,000 images in the datasets, so the RS attacker does not use up all the 50,000 query budget.

the effectiveness of different attacks given 500, 5,000, or 50,000 seed samples, which are 1%, 10%, or 100% of the query budget respectively. In particular, we run the random sampling (RS) attack as the baseline attack, since the attacker only queries the seed samples and does not use any data augmentation technique. We evaluate the gain of extraction accuracy of JB-top3 and JBA-PGD attacks over the RS baseline, shown in the Gain columns in Table 1.

Our first observation is, JBA-PGD and JB-top3 attacks are not very effective compared to RS if the attacker already has access to 5,000 images (10% of training samples) from the victim's training set. By examining the row of CIFAR10 in Table 1, we notice that when the attacker has 5,000 seed samples, the gain of JBA-top3 and JBA-PGD attacks is only 2% improvement over the extraction accuracy of RS. However, if the attacker has only 500 seed samples, the extraction accuracy gain of JBA-PGD increases to 7%, which is more significantly than the 5,000 seed samples scenario. Therefore, for the rest of the paper, we decide to focus on the threat model where the attacker has access to only 500 of seed samples.

We are also interested in the performance of JBA attacks when the distribution of the seed query set is different from the victim's training set. In practice, a realistic attacker may not have access to any of the original training samples, but instead may be able to collect publicly available datasets from a different distribution. Different rows of Table 1 show the extraction accuracy of different attackers when they have access to the seed samples from the corresponding datasets. Among the 8 datasets other than CIFAR10, CINIC10 is most similar to CIFAR10, so the extraction accuracy is the highest. Even though TinyImageNet is larger than CINIC10, the extraction accuracy using seed samples from TinyImageNet is lower than using CINIC10, this means the similarity to the victims dataset is very important to extract an accuracy model. Across all seed query sets, the extraction accuracy of JBA-PGD is consistently higher than JB-top3, when the attacker has access to only 500 seed samples.

As reference, we show the extraction accuracy of RS when the attacker has 50K seed samples in the last column of Table 1, which simulates the situation where the attacker has access to a large amount of natural images. Using seed samples from all but three datasets (Indoor67, CUBS200, and SVHN), the RS attack outperforms JBA attacks. Note that Indoor67, CUBS200, and SVHN have less than 50K data in their training set. Therefore, the RS attacker only queries all available samples from these dataset, without consuming the full 50K query budget. However, JBA attacks can make additional queries and thus outperforms the RS attack using these three datasets.

Summary: We will focus on the threat model where the attacker has access to only 500 seed samples, since the extraction accuracy gain for JBA attacks in CIFAR10 is more significant in this setting. Moreover, JBA-PGD attack is more effective than JB-top3 given access to different types of seed query sets.

4.3 Training SEAT

Table 2: Against the SEAT detector with four different similarity encoders, we show the number of accounts required for the attacker to extract the model. The best SEAT detector is trained using PGD attacks with $\epsilon = 8/255$.

JBA-PGD	**RT**	**SEAT (Ours) with ϵ**		
ϵ		8/255	16/255	24/255
8/255	13	**65**	43	28
16/255	9	**52**	32	22
24/255	7	**41**	25	18
32/255	6	**33**	20	15
40/255	4	**28**	17	12

To train the similarity encoder in our SEAT detector, we experiment with different methods to generate similar pairs and choose

the strongest detector. We train three similarity encoders by adversarial training with PGD attack for 30 epochs: for each natural image, PGD generates a positive sample by perturbing it for 40 iterations with random start and $\epsilon = 8/255$, $16/255$, or $24/255$. Moreover, following the Stateful Detector [7], we train a fourth similarity encoder using random transformations (RT), by slightly distorting and transforming images to generate positive pairs. The details of the distortion parameters are in Table 12 in Appendix B. Then, we tune the similarity threshold δ to meet the requirement of 0.01% false positive rate on benign queries, using both the training and test samples in CIFAR10 and CINIC10.

The best detector should be able to cancel the most victim's accounts. Therefore, we run the JBA-PGD attacks given 500 CIFAR10 images as the seed query set, with $\epsilon = 8/255, 16/255, 24/255, 32/255$, and $40/255$. Table 2 shows the number of accounts the attacker needs to extract the model when four different SEAT detectors are deployed. Each column in the table corresponds to a different training method. When the similarity encoder is trained by PGD with $\epsilon = 8/255$, the attacker needs the highest number of accounts, no matter what ϵ the attacker uses. For the rest of the paper, we will use the SEAT detector trained by PGD with $\epsilon = 8/255$ in the experiments.

4.4 False Positive Rate

In table 3, there are 13 experiments, each with benign queries from a different dataset, and false positive rates are estimated by the percentage of benign queries raising alarms. 12 of these datasets (all except CIFAR10) are hold-out sets, and they have not been used for training the defense. Table 3 shows that the false positive rate (FPR) of our SEAT detector is very low when benign users send queries from different query sets. Across 13 different datasets, the FPR of benign queries are all under 0.05%. In particular, to show that our detector maintains low FPR even when a user is sending naturally similar queries, we select 4 datasets: KaggleFrames[5] for video frames, GTSRB[17] for traffic sign recognition, LFW[18] for face recognition, and VGGFlower17 [6] for flower recognition. KaggleFrame is a video frame dataset, and the other three datasets have naturally similar samples within the same class/category (e.g., face and traffic sign recognition). The FPR for these datasets remains low. In Section §4.8, we will show that we achieve better FPRs in comparison to existing detection schemes. For example, the FPR of KaggleFrames is 0.05% for our detector, but 0.9% FPR for PRADA and 0.2% of FPR for the stateful detector (Table 9).

4.5 Effectiveness of SEAT Detector

Without deploying the SEAT detector, the attacker only needs one account to query the victim's prediction API. To evaluate the effectiveness of SEAT detector, we measure the number of fake accounts an attacker needs to query the protected model, which increases the cost for non-adaptive attackers to extract a model.

Table 4 summarizes the number of fake accounts and the extraction accuracy for different variants of JBA attackers after deploying the SEAT detector. The JBA attackers have access to 500 seed samples from CIFAR10. The JB-top3 attacker and JBA-CW_ℓ_2 attackers need more than 100 accounts, and the extraction accuracy is lower than models obtained by the JBA-PGD attackers. This is consistent with our finding in Section §4.2 that JBA-PGD outperforms JB-top3.

For different JBA-PGD attackers, Table 4 shows that the number of fake accounts decreases as ϵ increases, since a larger ϵ makes the augmented samples dissimilar. However, as ϵ increases, the extraction accuracy first increases and then decreases, and $\epsilon = 24/255$ has the highest extraction accuracy. Given this result, we choose $\epsilon = 24/255$ as the setup for JBA-PGD attack in the rest of the paper.

Although we train the similarity encoder using PGD attacks on the victim's CIFAR10 training set, SEAT can also detect JBA

Table 3: False Positive Rate of SEAT Detector

Query Set	FPR
CIFAR10	0.012%
TinyImageNet	0.007%
ImageNet1k	0.010%
CIFAR100	0.013%
SVHN	0.026%
CINIC10	0.009%
Indoor67	0.006%
CUBS200	0.017%
Caltech256	0.027%
KaggleFrames	0.050%
GTSRB	0.010%
LFW	0.012%
VGG-Flower17	0.037%

Table 4: The number of accounts needed by different variants of JBAs (with 500 CIFAR10 seed samples) to extract the CIFAR10 model protected by the SEAT Detector, with the corresponding extraction accuracy.

JBA	# Accounts	Ex. Acc.
JB-top3	122	69%
JBA-CW_ℓ_2	109	62%
JBA-PGD $\epsilon = 8/255$	65	73%
JBA-PGD $\epsilon = 16/255$	52	74%
JBA-PGD $\epsilon = 24/255$	**41**	**75%**
JBA-PGD $\epsilon = 32/255$	33	72%
JBA-PGD $\epsilon = 40/255$	28	71%

Table 5: The number of accounts and extraction accuracy for JBA-PGD attack ($\epsilon = 24/255$) to extract the model protected by SEAT Detector, given access to different types of seed set.

Seed Set	# Accounts	Ex. Acc.
CIFAR10	41	75%
TinyImageNet	29	62%
ImageNet1k	35	58%
CIFAR100	43	61%
SVHN	48	33%
CINIC10	30	70%
Indoor67	29	55%
CUBS200	41	34%
Caltech256	65	61%

[5]KaggleFrames: https://www.kaggle.com/akshaybapat04/frames-from-video
[6]VGGFlower17: https://www.robots.ox.ac.uk/~vgg/data/flowers/17/

attacks that use seed samples from other datasets. Table 5 shows the number of accounts required by the SEAT detector for different types of seed sets, including 8 datasets other than CIFAR10. The results show that we increase the accounts needed by the attacker by at least 29 times, to as high as 65 times.

4.6 Adaptive Attack Evaluation

An effective and robust defense scheme should work even when the attacker has full knowledge of the defense. In this section, we try to bypass our SEAT detector using the two adaptive attacks introduced in Section §3.3: *query filtering* (QF, Section §3.3.1) and *query blinding* (QB, Section §3.3.2). The adaptive attackers have access to 500 CIFAR10 seed samples as the seed query set.

Table 6: Number of fake accounts for adaptive QF and QB attackers to extract a model protected by SEAT detector, along with the extraction accuracy. Compared to the non-adaptive attacker, we make the best numbers of adaptive attackers in bold.

Adaptive Schemes	Strategies	# Accounts	Ex. Acc.
Non-adaptive	N/A	41	75%
Query Filtering	VGG16 + *CINIC10*	42	75%
	5-layer + *CIFAR10 seed*	**33**	**75%**
Query Blinding	Crop	21	71%
	Brightness	38	61%
	Scale	24	73%
	Rotate	41	75%
	Contrast	14	61%
	Uniform	64	62%
	Gaussian	65	60%
	Translate	**23**	**73%**
	Auto-encoder	31	60%

The QF attacker trains a surrogate similarity encoder to locally filter queries before sending them to the victim model. We experiment with two configurations of *encoder architecture + similarity training set*. The first configuration uses VGG16 with batch norm, the same as the victim's architure, and the attacker adversarially trains the simialrity encoder using surrogate dataset CINIC10 that is close to CIFAR10. In the second figuration, the attacker uses a 5-layer CNN architecture as in [7], and adversarially trains the similarity encoder using 500 CIFAR10 seed samples. Table 6 shows that the "5-layer + CIFAR10 seed" QF attacker is more effective than the other configuration, since the samples used for training is part of the victim's training set, and the small CNN architecture can learn an effective surrogate similarity encoder given the limited number of samples. However, SEAT detector still requires this QF attacker to use 33 fake accounts in order to extract a model.

We also experiment with the QB attack using a variety of random transformation strategies listed in Table 6. The details of the distortion parameters are in Table 12 in Appendix B. Among these QB attackers, contrast and crop transformations make the attacker use the fewest accounts, 14 and 21, respectively. However, their extraction accuracy is low, only 61% and 71%, respectively. The QB attacker using translate strategy achieves the second highest extraction accuracy, 73%, and reduces the number of accounts from 41 to 23, compared to the non-adaptive attacker. Therefore, we conclude that, against the QB attacker, the SEAT detector still requires the attacker to use 23 accounts to extract an accurate model.

In appendix table 11, we demonstrate that our detector still works when the attacker is able to collect 5,000 natural images from CIFAR10, which implies less chance to generate similar pairs.

4.7 Detecting DFME Attack

Our SEAT detector can also detect queries generated by the DFME attack. Following the setting in previous work [45], we grant 20M queries to DFME attacker. Also, we observe that the extraction accuracy of DFME converges around 20M queries. While its extraction accuracy is much higher than JBAs with 500 seed images, DFME needs way more queries than JBAs and are very easy to be detected by SEAT detector, shown in Table 8.

4.8 Comparison with Existing Defenses

In this section, we compare our SEAT detector to four different baseline detectors to demonstrate that our detector is more effective and more robust.

The first baseline is a random detector, where we cancel a user's account for every 10,000 queries, which allows us to keep the false positive rate around 0.01%. Since the attacker is granted 50,000 query budget, one baseline detector will detect the attack 5 times in total, which means that the attacker needs 5 fake accounts to successfully extract a model. We mark this baseline detector as *Random* in tables 7 and 9.

In addition, we choose the following three baseline detectors: out-of-distribution detector (OOD)[1], stateful detector (SD)[7], and PRADA [22]. Among existing model extraction defenses described in Section §2.6 and Section §2.7, the OOD detector and PRADA can work under the account setting, i.e., we can set up the defense to detect whether an account holder is adversarial and cancel the account after each detection. The other baseline, SD, does not detect model extraction attacks, but it detects adversarial examples. However, SD also tracks the historical queries and compare the similarity among the queries. Therefore, we choose to compare our SEAT detector against SD as well. We tune the thresholds for theses detectors on the same tuning set as ours (CIFAR10 and CINIC10) to have FPR 0.01%.

We experiment with three JBA-PGD attacks ($\epsilon = 24/255$) that use 500 seed images sampled from CIFAR10, TinyImageNet, and CINIC10, respectively. This simulates the threat models where the attacker has access to subset of victim's training set (CIFAR10), a more general dataset (TinyImageNet), and a highly similar surrogate dataset (CINIC10). In Table 7, we show the number of accounts for non-adaptive attacks in the "Vanilla" columns. The OOD and SD detectors are not effective, and they underperform the random baseline. On the other hand, both PRADA and SEAT detector can increase the number of accounts by order of magnitude.

Furthermore, we run adaptive query blinding (QB) attacks with random translation to compare PRADA against SEAT detector. As shown in the "Adaptive" columns in Table 7, the query blinding attack reduces the number of accounts for PRADA to at most 5, but it still needs 16 to 23 accounts for SEAT detector. PRADA compares images by ℓ_2-distance, which is highly vulnerable to query blinding

Table 7: Number of attacker accounts against baseline detectors and our SEAT detector. "Vanilla": non-adaptive attack. "Adaptive": adaptive query blinding attack with random translation.

	Random	OOD[1]	SD[7]	PRADA[22]		SEAT (Ours)	
Seed Images	Vanilla	Vanilla	Vanilla	Vanilla	Adaptive	Vanilla	Adaptive
CIFAR10	5	5	1	203	2	41	**23**
TinyImageNet	5	5	1	218	5	29	**19**
CINIC10	5	1	1	66	1	70	**16**

Table 8: Number of accounts for non-adaptive and adaptive DFME (20M Queries) attackers to extract a model protected by SEAT.

Adaptive Scheme	Strategies	# Accounts	Ex. Acc.
Non-adaptive	N/A	57010	89%
Query Filtering	5-layer + *CIFAR10 seed*	57601	89%
Query Blinding	Uniform	57707	87%
	Gaussian	58443	88%
	Scale	57014	87%
	Brightness	58042	86%
	Rotate	57002	89%
	Contrast	57826	73%
	Crop	57013	86%
	Translate	56981	88%

attacks, (as shown in table 7): it is easy to construct a similar image with a large ℓ_2-distance (e.g., brighten the image). In contrast, SEAT uses a similarity encoder to measure image similarity, which is robust against query blinding attacks. Overall, when the seed images are from CIFAR10, the SEAT detector increases the number of accounts needed by the attacker by 23 times (from 1 to 23); and given TinyImageNet seed images, the SEAT detector increases the number of accounts by 3.8 times (from 5 to 19).

Table 9: FPRs of the random detector, three existing defenses (PRADA, OOD, SD), and our SEAT detector.

Query Set	Random	PRADA[22]	OOD[1]	SD[7]	SEAT (Ours)
CIFAR10	0.01%	0.000%	0.013%	0.008%	0.012%
TinyImageNet	0.01%	0.000%	0.005%	0.006%	0.007%
CINIC10	0.01%	0.000%	0.009%	0.009%	0.010%
KaggleFrames	0.01%	0.900%	0.000%	0.200%	0.050%
GTSRB	0.01%	0.004%	0.002%	0.004%	0.010%
LFW	0.01%	0.012%	0.002%	0.000%	0.012%
VGG-Flower17	0.01%	0.074%	0.000%	0.000%	0.037%

Table 9 shows the FPR of different detectors when a user queries benign images from different datasets. The underlined numbers suggest that PRADA and SD become too sensitive if a benign user sends queries from KaggleFrames video frame dataset, where their FPR is much higher than our SEAT detector.

4.9 Ablation Study

We conduct an ablation study to demonstrate that our similar-pair-based detection scheme outperforms the cluster-based detection scheme. In this experiment, both detectors apply the same similarity encoder, and the only difference is their detection schemes. Table 10 summarizes the results and show that similar-pair-based scheme is about 2× better than cluster-based.

Table 10: Number of attacker accounts against cluster-based v.s. pair-based detection scheme in SEAT detector

	Cluster-based		Pair-based (Ours)	
	Vanilla	Adaptive	Vanilla	Adaptive
CIFAR10	13	10	41	**23**
TinyImageNet	9	7	29	**19**
CINIC10	12	7	70	**16**

5 LIMITATIONS

5.1 False Positive Rate

Although our SEAT detector achieves 0.01% false positive rate, it is worth mentioning that in reality, the number of benign queries is much larger than the adversarial ones. Therefore, a 0.01% FPR might still be too high for practical deployment because of the base rate problem.

5.2 Other Adaptive Attacks

An anonymous reviewer suggested *Query Partitioning* (QP), an adaptive attack against our detector: the attacker avoids similar pairs sent from the same account by carefully partitioning queries and assigning them to different accounts. Suppose the attacker has m seed images, is granted a budget of B queries, and has N_A accounts. If $N_A = B/m$, then in each round, the attacker could sample m images, augment them, and query them, using a different account for each round. Because no two images derived from the same seed image are queried by the same account, SEAT is unlikely to detect this attack. For the parameters we focus on in this paper, this would require $N_A = 50000/500 = 100$ accounts, which is more than a query blinding attack—thus this specific query partitioning attack is less effective than other attacks. We leave it to future work to determine whether some other partitioning might be more effective. For instance, it might be more effective to use account i to query images from rounds $i, i + 20, i + 40, \ldots$.

5.3 Pair Search Runtime

One performance bottleneck of the SEAT detector is the running time to search for similar pairs. The running time for each query is linear in the number of previous queries from the same account, but not the total number of previous queries. This might be acceptable for accounts that don't make too many queries, and perhaps high-traffic accounts could be subject to other mitigations (e.g., additional identity verification). It might be possible to speed up similar-pair search using locality-sensitive hashing; we have not explored this direction.

6 CONCLUSION

In this paper, we have presented SEAT detector, a new detector for detecting Jacobian-based model extraction attacks. SEAT detector consists of a Similarity Encoder trained by Adversarial Training and similar-pair-based detection scheme. Our experiments have shown that SEAT outperforms existing detection schemes while maintaining high utility for benign users.

7 ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive and valuable feedback. This research was supported by generous gifts from Open Philanthropy, Google, and Berkeley Artificial Intelligence Research (BAIR).

REFERENCES

[1] Buse Gul Atli, Sebastian Szyller, Mika Juuti, Samuel Marchal, and N. Asokan. 2020. Extraction of Complex DNN Models: Real Threat or Boogeyman? arXiv:1910.05429 [cs.LG]

[2] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2018. CSI Neural Network: Using Side-channels to Recover Your Artificial Neural Network Information. arXiv:1810.09076 [cs.CR]

[3] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. arXiv:1712.04248 [stat.ML]

[4] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. 2020. Cryptanalytic Extraction of Neural Network Models. arXiv:2003.04884 [cs.LG]

[5] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. arXiv:1608.04644 [cs.CR]

[6] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. 2020. HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. arXiv:1904.02144 [cs.LG]

[7] Steven Chen, Nicholas Carlini, and David Wagner. 2019. Stateful Detection of Black-Box Adversarial Attacks. arXiv:1907.05587 [cs.CR]

[8] Jacson Rodrigues Correia-Silva, Rodrigo Berriel, Claudine Santos Badue, Alberto Ferreira de Souza, and Thiago Oliveira-Santos. 2018. Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data. *2018 International Joint Conference on Neural Networks (IJCNN)* (2018), 1–8.

[9] Luke N. Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. 2018. CINIC-10 is not ImageNet or CIFAR-10. arXiv:1810.03505 [cs.CV]

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[11] John (JD) Douceur. 2002. The Sybil Attack. In *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)* (proceedings of 1st international workshop on peer-to-peer systems (iptps) ed.). https://www.microsoft.com/en-us/research/publication/the-sybil-attack/

[12] John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. 2012. Finite Sample Convergence Rates of Zero-Order Stochastic Optimization Methods.. In *NIPS*. Citeseer, 1448–1456.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML]

[14] Greg Griffin, Alex Holub, and Pietro Perona. 2006. Caltech256 Image Dataset. (2006). http://www.vision.caltech.edu/Image_Datasets/Caltech256/

[15] R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. 1735–1742. https://doi.org/10.1109/CVPR.2006.100

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV]

[17] 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 *International Joint Conference on Neural Networks*.

[18] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical Report 07-49. University of Massachusetts, Amherst.

[19] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box Adversarial Attacks with Limited Queries and Information. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 2137–2146. http://proceedings.mlr.press/v80/ilyas18a.html

[20] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167 [cs.LG]

[21] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2020. High Accuracy and High Fidelity Extraction of Neural Networks. arXiv:1909.01838 [cs.LG]

[22] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. 2019. PRADA: Protecting against DNN Model Stealing Attacks. arXiv:1805.02628 [cs.CR]

[23] Sanjay Kariyappa, Atul Prakash, and Moinuddin K Qureshi. 2021. Protecting {DNN}s from Theft using an Ensemble of Diverse Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=LucJxySuJcE

[24] S. Kariyappa and M. K. Qureshi. 2020. Defending Against Model Stealing Attacks With Adaptive Misinformation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 767–775. https://doi.org/10.1109/CVPR42600.2020.00085

[25] Alex Krizhevsky. 2012. Learning Multiple Layers of Features from Tiny Images. *University of Toronto* (05 2012).

[26] Ya Le and X. Yang. 2015. Tiny ImageNet Visual Recognition Challenge.

[27] Huiying Li, Shawn Shan, Emily Wenger, Jiayun Zhang, Haitao Zheng, and Ben Y. Zhao. 2020. Blacklight: Defending Black-Box Adversarial Attacks on Deep Neural Networks. arXiv:2006.14042 [cs.CR]

[28] Daniel Lowd and Christopher Meek. 2005. Adversarial Learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (Chicago, Illinois, USA) *(KDD '05)*. Association for Computing Machinery, New York, NY, USA, 641–647. https://doi.org/10.1145/1081870.1081950

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2019. Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083 [stat.ML]

[30] Sébastien Marcel and Yann Rodriguez. 2010. Torchvision the Machine-Vision Package of Torch. In *Proceedings of the 18th ACM International Conference on Multimedia* (Firenze, Italy) *(MM '10)*. Association for Computing Machinery, New York, NY, USA, 1485–1488. https://doi.org/10.1145/1873951.1874254

[31] Smitha Milli, Ludwig Schmidt, Anca D. Dragan, and Moritz Hardt. 2018. Model Reconstruction from Model Explanations. arXiv:1807.05185 [stat.ML]

[32] Seungyong Moon, Gaon An, and Hyun Oh Song. 2019. Parsimonious Black-Box Adversarial Attacks via Efficient Combinatorial Optimization. arXiv:1905.06635 [cs.LG]

[33] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. *NIPS* (01 2011).

[34] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2018. Knockoff Nets: Stealing Functionality of Black-Box Models. arXiv:1812.02766 [cs.CV]

[35] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2020. Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks. arXiv:1906.10908 [cs.LG]

[36] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. 2020. ActiveThief: Model Extraction Using Active Learning and Unannotated Public Data. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 01 (Apr. 2020), 865–872. https://doi.org/10.1609/aaai.v34i01.5432

[37] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. arXiv:1602.02697 [cs.CR]

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[39] A. Quattoni and A. Torralba. 2009. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 413–420. https://doi.org/10.1109/CVPR.2009.5206537

[40] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*. http://arxiv.org/abs/1707.04131

[41] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. 2020. Foolbox Native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX. *Journal of Open Source Software* 5, 53 (2020), 2607. https://doi.org/10.21105/joss.02607

[42] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet. 2016. Adversarial Manipulation of Deep Representations. arXiv:1511.05122 [cs.CV]

[43] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs.CV]

[44] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs. arXiv:1609.02943 [cs.CR]

[45] Jean-Baptiste Truong, Pratyush Maini, Robert J. Walls, and Nicolas Papernot. 2021. Data-Free Model Extraction. arXiv:2011.14779 [cs.LG]

[46] Daniel Ponsa Vassileios Balntas, Edgar Riba and Krystian Mikolajczyk. 2016. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, Edwin

R. Hancock Richard C. Wilson and William A. P. Smith (Eds.). BMVA Press, Article 119, 11 pages. https://doi.org/10.5244/C.30.119

[47] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. 2011. *The Caltech-UCSD Birds-200-2011 Dataset.* Technical Report CNS-TR-2011-001. California Institute of Technology.

[48] Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. 2020. Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. In *29th USENIX Security Symposium (USENIX Security 20).* USENIX Association, 2003–2020. https://www.usenix.org/conference/usenixsecurity20/presentation/yan

[49] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y. Zhao, and Yafei Dai. 2014. Uncovering Social Network Sybils in the Wild. *ACM Trans. Knowl. Discov. Data* 8, 1, Article 2 (Feb. 2014), 29 pages. https://doi.org/10.1145/2556609

[50] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. 2020. CloudLeak: Large-Scale Deep Learning Models Stealing Through Adversarial Examples. Network and Distributed System Security Symposium. https://doi.org/10.14722/ndss.2020.24178

A ATTACKER STARTING FROM 5,000 SEED IMAGES

We use table 11 to show that our SEAT detector remains effective when the attacker collects 5,000 natural images for the seed image set and use the same amount of budget (50,000).

Table 11: Adaptive JBA-PGD with 5,000 CIFAR10 seed samples

Adaptive Schemes	Strategies	# Accounts	Ex. Acc.
Non-adaptive	N/A	29	87%
Query Filtering	VGG16 + *CINIC10*	28	86%
	5-layer + *CIFAR10 seed*	**25**	**85%**
	VGG16 + *CIFAR10 seed*	32	86%
Query Blinding	Crop	14	85%
	Brightness	27	82%
	Scale	14	85%
	Rotate	27	85%
	Contrast	12	82%
	Uniform	56	82%
	Gaussian	54	82%
	Translate	**15**	**85%**
	Auto-encoder	22	83%

B DISTORTION PARAMETERS OF RANDOM TRANSFORMATION

Table 12: Random Transformation Parameters

Transformation	Torchvision	Parameter
Crop	*RandomResizedCrop*	scale=(0.96, 1)
Brightness	*ColorJitter*	brightness=0.09
Scale	*RandomAffine*	scale=(0.83, 1.17)
Rotate	*RandomRotate*	degrees=15
Contrast	*ColorJitter*	contrast=0.55
Translate	*RandomAffine*	translate=(0.05, 0.05)
Uniform noise	N/A	range=(-0.064, 0.064)
Gaussian noise	N/A	(mean,std)=(0,0.095)

Chen *et al.*[7] generate positive sample in equation (3) by random transformations. In our experiments, we use Torchvision[30] implementation of these random transformation, and an transformation will be randomly sampled to transform an images. For Uniform and Gaussian noise, we use PyTorch implementation of random number generator of corresponding distributions with necessary scaling and shifting. In table 12, the first column is the type of transformation, the second column is the implementation name in Torchvision, and the last column is how we set the parameter of these random transformation.