Towards Characterizing Model Extraction Queries and How to Detect Them

Zhanyuan Zhang
Yizheng Chen
David A. Wagner

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-126
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-126.html

May 14, 2021

Towards Characterizing Model Extraction Queries and How to Detect Them

by Zhanyuan Zhang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

D. Wagner

Professor David Wagner
Research Advisor

5/14/2021

(Date)

* * * * * * *

Professor Michael Mahoney
Second Reader

5/14/21

(Date)

Towards Characterizing Model Extraction Queries and How to Detect Them

Zhanyuan Zhang
University of California, Berkeley

Yizheng Chen
Columbia University

David Wagner
University of California, Berkeley

***Abstract*—Machine Learning as a Service (MLaaS) has become popular in cloud services as Deep Neural Networks (DNNs) are demonstrating high-performance in many domains and as the rapid growth in cloud computing. Meanwhile, developing enterprise MLaaS remains costly since training machine learning models typically requires large-scale data collection and labeling. However, researchers have shown that model extraction attacks are able to *steal* functionality of models deployed on Cloud only through black-box access to victim's models and sending adversarial queries to application programming interface (API). This information leakage indicates potential threats to protecting enterprise machine learning models as a part of intellectual property. In this paper, we present two lines of our research on model extraction attacks: characterizing adversarial queries and building detectors against them. In our first line of research, we find that although adversarial queries help adversary explore victim's decision regions to some extent, they fail to extract properties of decision boundaries, which is most of the existing algorithms claim to be capable of. In our second line of research, we propose two ways to detect Jacobian-based and Data-free model extraction attacks: 1) a similarity-based detector to show the possibility of building a robust detector against model extraction attacks by adopting detectors for adversarial examples, and 2) a VAE-based detector that uses Variational Autoencoder to estimate whether queries are benign or not.**

***Index Terms*—model extraction/stealing, black-box attacks, adversarial machine learning, MLaaS, intellectual property**

I. Introduction

Deep Neural Networks (DNNs) has seen tremendous success in many domains. Moreover, the rise of cloud platforms has made Machine Learning as a Service (MLaaS) (e.g., Microsoft Azure, Google Cloud Platform, Amazon Web Services, Clarifai) more accessible. However, MLaaS faces a unique challenge: model extraction attacks (MEA) can steal the functionality of the model from a cloud platform, by querying the model on many different inputs. This poses an business risk to MLaaS service providers, since proprietary models are part of their intellectual property.

Model extraction attacks enable an adversary to construct a surrogate model that performs similarly to the service provider's model (the victim model). These attacks work by sending queries to the ML prediction API and utilizing information leaked via the API's outputs. Researchers have shown that model extraction attacks can be successful in a black-box setting, without knowledge of victim's training set, model architecture, and other hyperparameters. In this paper, we mainly focus on learning-based approaches to MEA [1]. Adversaries using this approach will the API to build a substitute dataset and train their surrogate model on it. There are three main categories of learning-based MEA: Jacobian-based Augmentation (JBA), Data-free (DF), and Active Learning (AL) attacks (section §III). Briefly speaking, a JBA attack starts with a small dataset of samples, then augments these samples by constructing points that are near the decision boundary on the surrogate model. A DF attack generates a substitute dataset using a generative model, without needing any initial dataset. An AL attack constructs its substitute dataset using active learning.

In our first line of research, we investigate MEA attacks. Many attacks assume that attackers have access to a dataset of training images, but this assumption may not always be realistic. We investigate the effectiveness of different MEA attacks, under different assumptions about what kind of dataset the attacker has access to. Prior work on JBA attacks [2], [3] assumes that adversaries have a subset of victim's training set, but in practice, training sets are often proprietary and may not be available to the adversary. Therefore, we measure the effectiveness of JBA attacks when the attacker has access to some other publicly accessible dataset, but not the victim's training set. We also compare JBA attacks to the other kinds of MEA attacks. Our experiments show that JBA is useful for attackers: JBA's augmentation improves the substitute model's accuracy from 2–23 percentage points over other attacks, with the advantage most pronounced when the attacker can only collect a limited amount of data; in contrast, in this regime there is not sufficient data for an AL attacker to learn efficient sampling strategies, or for a DF attacker to learn how to generate data similar to the victim's proprietary data. These results show that JBA attacks can be a significant threat to MLaaS service providers.

We also investigate the reasons why JBA attacks succeed. Prior works on JBA attacks [2]–[4] argue that their approaches extract models by generating instances that are near or just barely cross the victim model's decision boundary, helping the attacker learn exactly where victim model's decision boundary lives. We empirically test this hypothesis by measuring the proportion of crossed samples and the predicted confidence of the victim's model to characterize and locate adversarial queries. The proportion of crossed samples measures how many augmented samples land on the opposite side of the decision boundary as the original sample. The predicted confidence (top-1 softmax) is a proxy for the sample's distance to the closest decision boundaries (the lower the closer). In our

experiments (section §IV), we find that JBA attacks are not effective at generating queries that cross the victim model's decision boundary, so previous explanations about how JBA attacks work may not be valid in general. In addition, we find that DF attack, although it needs $400\times$ more queries than JBA to reach similar performance, can effectively generate queries that explore a wide range of the victim model's decision boundary.

In our second line of research, we propose defenses against MEA attacks. We explore two ways to build a detector that can detect a MEA in progress and terminate the adversary's accounts before the attack is successful. We build on prior stateful defenses for detecting black-box attacks ([2], [5], [6]), which work by tracking the sequence of queries from an account and checking if they follow a pattern that is characteristic of known attacks. We extend these methods to build a robust stateful detector against JB and DF MEA attacks, while maintaining API's utility so that benign users will not be affected. In particular, we adapt the Stateful Detector by Chen *et al.* [5] to build a similarity-based detector against JBA and Data-free MEA. Following [5], we test the similarity-based detector against adaptive Query Blinding attacks (QB), and show that our detector is still robust to QB: QB reduces the number of fake accounts an adversary would need by about $2\times$, but even with the best QB attack, the adversary would still need 8 fake accounts to mount a successful MEA.

We further introduce a novel adaptive attack, the Query Filtering attack (QF). The QF attack is designed specifically to evade stateful detectors. In a query filtering attack, the attacker locally emulates the behavior of the detector to identify which queries might raise alarms, and then filters those out (i.e., only queries the API on samples that are predicted to avoid raising an alarm). Our best QF attack is able to use $0.19\times$ less fake accounts comparing to a non-adaptive attack, which can be achieved when the attacker has access to part of victim's training set.

We propose a second stateful VAE-based detector that monitors the accumulated likelihood of a sequence of queries. It uses a Variational Auto-encoder to estimate how likely a query is benign. The defender terminates an account when it sees many queries that are unlikely to be benign. We find that although our VAE-based detector outperforms our similarity-based detector on the dataset it was specifically trained on, it is prone to false positives, particularly if honest users send queries from other datasets. When honest users may query a broad variety of images, such as images from other distributions or datasets, our VAE-based detector either is not sensitive to adversarial queries, or it is too sensitive to benign queries.

In summary, we try to understand model extraction attacks from both the attacker's and defender's perspectives, and make the following contributions in this work:

1) Playing the role of attacker, we find that:
   - Jacobian-based model extraction attack (JBA) has an advantage over Active Learning and Data-free attacks when the adversary can only collect a limited amount of natural data for sending adversarial queries.
   - Prior works on JBA have claimed that JBA generates queries that cross the victim model's decision boundary, or lie close to it. We disprove those claims.
   - Complexity in query sets helps improve JBA to some degree, but to further improve attack performance, attackers need to choose query sets with a similar distribution to the victim's training set (more details in section §IV-C).
2) Playing the role of defender, we explore two ways of building detector for JBA attacks:
   - We adapt the *Stateful Detector* by Chen *et al.* [5] to MEA attacks and build a robust similarity-based detector that can detect a wide range of MEA attacks.
   - We introduce *query filtering*, an adaptive attack where the adversary filters out queries that may cause the detector to raise an alarm.
   - We design a VAE-based detector and identify some challenges with such a defense.

The rest of this paper will be structured as follow: Section §II introduces the threat model that we focus on throughout this paper, and section §III gives an overview on most of the existing model extraction attack. We explore and characterize these attacks in section §IV. In section §V, we introduce existing defenses/detection schemes against model extraction attacks. We propose novel detection schemes in section §VI and adaptive attacks in section §VII. Finally, we evaluate our detectors in section §VIII.

II. Problem Statement

In this section, we state the threat model that we apply to study model extraction attacks (MEAs) in this paper. For simplicity, F_A denotes adversary's surrogate model, and F_V denotes victim's secret model.

A. Adversaries' Goal

In practice, adversary' goal is to obtain a copy of surrogate model that perform well on victim's task. In our experiments, we simulate this by evaluating the accuracy of F_A on victim's test set, to which we refer as *extraction accuracy*, although in reality, adversary often has no access to this test set.

As summarized by Jagielski *et al.* [1], adversary can have other objectives while running model extraction and other similar black-box attacks: some looking for high prediction agreement (making the same predictions regardless of correctness) between F_V and F_A ([1], [2], [4], [7], [8]); others aim for recovering victim's weights in surrogate model exactly ([7], [9]–[12]).

B. Adversary's Capabilities

In this paper, we focus on *learning-based* attackers. As described in [1], learning-based attackers send queries and ask F_V to label them. Those queries and F_V's labels form a substitute dataset, where the attackers train their surrogate model to achieve good performance on victim's task. Therefore, any attacks that require more than F_V's outputs are out of our scope. For instance, *Functionally Equivalent Extraction* attacks

require access to F_V's ReLU function [1] or the prediction confidence [9], CSI NN in [12] needs power side channel, and the model reconstruction in [11] require F_V to return gradient.

Query Sets: The attacker can collect a subset (*e.g.* 1% to 10%) of F_V's unlabeled data $\mathcal{X}_V$ in training set. When the attacker has a subset of victim's training set, we make sure that the seed query set $\mathcal{X}_0$ is balanced, which means each class has the same amount of data. This is reasonable since the attacker knows the number of classes in victim's task. Besides, the adversary is able to collect data from all the publicly accessible datasets. To simulate this, we choose datasets that cover the spectrum of complexity (relative to victim's training set) as much as possible.

Black-box access: The adversary only has black-box access to victim's model – adversary can send any data to the prediction API provided by the victim, and observe the outputs. We simulate this by only allowing surrogate models to use their queries and outputs from F_V during its training process.

Hard-label: As for output granularity of the prediction API, we believe that it is close to real-world practice to focus on *hard-label* setting, where prediction API outputs its prediction label but not its confidence scores. We belive it is also fair to let the adversaries know how many candidate classes in victim's task. These are simulated by letting F_V output an one-hot encoding vector, which has 1 at the index of the predicted class, and 0's in the rest of positions. However, for evaluation purpose, we relax this constraint for Data-free attack, since adversary needs F_V to return logits (F_V output without softmax) so that the training of generative model is possible (more details in section §III-B).

Sybil accounts: As many prior works ([5], [6], [13], [14]) have discussed, it is totally fair to assume that adversaries may perform attacks through many *Sybil* accounts or IP addresses.

C. Defender

Who's the victim: The defender is protecting a proprietary image classifier F_V. Clients must interact with F_V through an API from the service provider.

Monitoring queries: The defender monitors queries sent to F_V and is able to keep track of some metric or history about the queries, hence our defender is stateful.

Defender's goal is to detect and terminate accounts sending queries for Jacobian-based model extraction attacks, without affecting the usage of benign users. Every time the detector raises an alarm, we consider it as terminating the attacker's account, but allow the attack to continue as if it's under a new account until the adversary exhausts the query budget. Note that since we allow users, regardless of whether they are benign or malicious, to send queries from any sources different from victim's, we consider all natural images benign. Hence, the detectors we propose are only for finding out synthesized data, but not those natural ones. More specifically, this implies that our detectors work against Jacobian-based augmentation and Data-free attacks, but not Active Learning attacks, since the former two attacks synthesize data, but the latter one queries all the natural images, which should be *benign* by our design (mode details in section §III).

Detection Metrics:

- *Number of accounts*: We count the number of accounts detected by the defender during a MEA. This simulates the situation (*e.g. Sybil* attack) where the attacker creates a new account once the previous ones get detected and terminated by the defender.
- *False Positive Rate (FPR)*: The defender should not affect the utility of API. The usage of FPR is two-fold: 1) we tune our detectors such that it has 0.01% FPR on tuning set. This implies that we expect in practice, a benign user sending 10,000 queries is very likely trigger a false positive alarm; 2) to evaluate the FPR of our detectors, we send natural images from a different distribution than the tuning set's.

III. Model Extraction Attacks

In this section, we give an overview on most of the existing model extraction attack algorithms in prior works. We additionally propose an attack based on binary search.

A. Jacobian-based Augmentation Attacks

Many existing model extraction attacks fall into the category of Jacobian-based augmentation attack (JBA) ([2] [3] [4] [7]). We will first describe the Jacobian-based augmentation attack (JBA) algorithm, then discuss some variants of JBA, and lastly propose our JBA attack variant (JBA-BS) based on line search.

1) Attack Algorithm: JBA starts with a seed query set $\mathcal{X}_0$: [2] and [3] start with some subset of victim's training set, while our threat model allows adversary to start with any other public accessible datasets. In general, adversary asks victim to label their query set and applies some *augmentation algorithms* to augment the query set. Algorithm 1[1] presents more details in the procedure above.

Augmentation algorithm is essential to JBAs since adversary relies on augmented data to explore some important properties (*i.e.* decision boundary) of victim's model. Since adversary has only black-box access, adversary use their working model F_A as a proxy to victim's model.

[1]Follow the official implementation: https://github.com/tribhuvanesh/prediction-poisoning/blob/master/defenses/adversary/jacobian.py

Algorithm 1 Jacobian-based Augmentation Attacks

Input: Seed query set $\mathcal{X}_0 = \{x_1, x_2, \ldots, x_m\}$; Victim's Model F_V; Query budget B; n training epochs for each augmentation round; N training epochs after all rounds; Augmentation Algorithm Aug

Output: Trained Substitute Model F_A

Initialisation : $X_Q \leftarrow \mathcal{X}_0$; Initialize F_A randomly or by loading pre-trained weights; $\mathcal{D}_T \leftarrow \emptyset$

1: $Y_Q \leftarrow F_V(X_Q)$ (Send queries to F_V)
2: $\mathcal{D}_T \leftarrow \mathcal{D}_T \cup \{(X_Q, Y_Q)\}$
3: Train F_A on $\mathcal{D}_T$ for n epochs
4: **while** Number of queries < B - m **do**
5: $X_Q \leftarrow Aug(\mathcal{D}_T, F_A)$
6: $Y_Q \leftarrow F_V(X_Q)$ (Send queries to F_V)
7: $\mathcal{D}_T \leftarrow \mathcal{D}_T \cup \{(X_Q, Y_Q)\}$
8: Reinitialize F_A and train it on D_T for n epochs
9: **end while**
10: Reinitialize F_A and train it on D_T for N epochs
11: **return** F_A

One can abstract augmentation algorithm as a procedure of perturbing the original data x through the guide of Jacobian calculated from some loss function $\mathcal{L}(x, F_A(x))$, as shown in Algorithm 2:

Algorithm 2 Augmentation Algorithm

Input: Original data x; Adversary's Model F_A; Objective 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'

2) Attack Variants: Different Jacobian-based extraction algorithms mainly differ in their loss functions $\mathcal{L}$. Most of the existing works design their loss functions based on the loss functions for crafting adversarial examples:

Papernot *et al.* [4] and Juuti *et al.* [2] propose their loss functions similar to the one used by targeted/non-targeted *Projective Gradient Descent Attack* (PGD) [15], while Yu *et al.* [3] investigate *Carlini-Wagner* ℓ_2 *Attack* (CW_ℓ_2) [16] and *Feature Adversary Attack* [17].

Yu *et al.* [3] also propose *Feature Fool* (FF), which is essentially the ℓ_2 distance between the original data x and augmented data x' plus a triplet margin loss [18] among the extracted features of x', x, and some input with targeted label x_t, which is mis-classified by F_A to some class t:

$$\begin{aligned} \min \mathcal{L}(x,x',x_t) = &\parallel x - x' \parallel_2 + \\ &\lambda \cdot \text{Triplet}(x, x', x_t) \\ &\text{such that } x' \in [0,1]^n \end{aligned} \tag{1}$$

and

$$\begin{aligned} \text{Triplet}(x, x', x_t) = \max\{(M + &\parallel \phi_k(x), \phi_k(x') \parallel_2 \\ &- \parallel \phi_k(x_t), \phi_k(x') \parallel_2), 0\} \end{aligned} \tag{2}$$

where $\phi_k(\cdot)$ denotes the feature vector extracted by the k^{th} layer of F_A, and $M = \alpha - \frac{1}{n_y^2 - n_y} \sum_{i,j \in y_s} \parallel \phi_A(x_i) - \phi_A(x_j) \parallel_2^2)$.

3) JBA-BS: Inspired by the line search in [7] and a way for characterizing DNN decision boundary by Karimi *et al.* [19], we propose JBA-BS. Given an example x_s whose original class is s and a targeted class t, JBA-BS first craft adversarial example $x_t = PGD(x_s, F_A, t)$. Then a binary search is done between x_s and x_t with update rules:

$$\begin{aligned} \text{Let } x_{\text{mid}} &\leftarrow \frac{x_s + x_t}{2} \\ \text{if } L_{F_A}(x_{\text{mid}}) = t &: \\ x_t &= x_{\text{mid}} \\ \text{else if } L_{F_A}(x_{\text{mid}}) = s &: \\ x_s &= x_{\text{mid}} \end{aligned}$$

and we terminate searching when any of the following stopping conditions is met:

$$\begin{cases} \text{Number of iterations exceeds } N_{\text{max}} \\ \parallel x_s - x_t \parallel_\infty < \epsilon \\ L_{F_A}(x_{\text{mid}}) \neq t \text{ and } L_{F_A}(x_{\text{mid}}) \neq s \end{cases}$$

In section §IV, we use PGD, CW_ℓ_2, and binary search to augment data and try to find out what happens as the Algorithm 1 (step 4 to step 8) iterates.

B. Data-free Extraction Attacks

Data-free extraction can be viewed as a special instance of JBA: instead of starting from some natural images, adversary synthesizes queries either by randomly sampling, or some generative models.

Tramèr *et al.* [7] purpose their algorithm for extracting shallow neural networks. They first generate some random data (say 25% of the budget), and then synthesize data by line search among these random data. In their implementation[2], line search is essentially binary search for finding the middle point between two samples.

Truong *et al.* [20] purpose their data-free model extraction *DFME* based on Generative Adversarial Networks (GAN) [21]. Given an objective function *L*, which evaluates the discrepancy between the performance of victim and adversary, adversary trains a generator G to generate queries that maximizes *L* from any data sampled from standard normal distribution, while adversary's substitute F_A model was trained to minimizes *L*:

$$\min_{F_A} \max_{G} \mathbb{E}_{z \sim \mathcal{N}(0,1)}[L(F_V(G(z)), F_A(G(z)))] \tag{3}$$

In their implementation[3], adversary alternatively trains G, for n_G iterations, and then F_V, for n_V iterations, until the budget is exhausted. Note that equation (3) involves F_V, but adversary only has black-box access to victim's model, so they need extra queries to approximate the gradient $\nabla_x F_V(x)$ by the Forward Differences method [22], specifically by

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

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

independently sampling m directions Z_i from the standard Normal distribution:

$$\nabla_{\text{FWD}} F_V(x) = \frac{1}{m} \sum_{i=1}^{m} \frac{F_V(x + \epsilon \cdot Z_i) - F_V(x)}{\epsilon} Z_i \quad (4)$$

In section §IV, we evaluate if the queries generated by DFME can effectively help adversary explore the decision boundary in victim's model. Usually, *data-free* means not using any natural images, but in section §VIII-A3 in order to demonstrate that our detector can also detect queries generated by GAN even with our adaptive attack *query filtering* (QF), we allow a adaptive DFME attacker to use the dataset that distributes most closely to victim's training (more details of QF in section §VII-B).

C. Active Learning Extraction Attacks

Researchers also purpose model extraction attacks based on Active Learning (AL). Unlike JBA, AL-based attacks usually do not generate data, rather, they learn some *sampling strategy* to sample those *informative* data from their query sets and ask victim to label these data to construct a *fake* dataset. Adversary then trains their substitute models on this dataset and expect their substitute models F_A achieve similar or even better performance as F_V. Adversary iteratively applies some sampling strategies to collect and send queries from publicly accessible sources, and the sampling strategies is trained to improve as iteration goes. Researchers have purposed several ways for constructing strategies and how to train them:

Correia-Silva *et al.* [8] purpose *CopyCat*, which collects queries by randomly sampling from datasets that may or may not be in the same problem-domain as victim. People often call this strategy *random strategy*, and it usually serves as a baseline for other active learning strategies, including the ones below.

Orekondy *et al.* [23] purpose *KnockoffNet* with *adaptive* strategies. Adaptive strategy is in a hierarchical structure like a tree, where data fall in different abstract levels (*i.e.* from top to bottom: animal $\rightarrow$ bird $\rightarrow$ sparrow). Starting from the top level, adversary traverses to the bottom with the branching probability specified by the adaptive strategy. The reward function for training this hierarchical strategy encourages high-confident sample for victim, diversity in sampling, and samples that reveal the difference between F_V and F_A. And they use gradient bandit algorithm [24] to train this adaptive strategy.

Pal *et al.* [25] purpose *Uncertainty*, *K-center*, *DFAL*, and *DFAL+K-center* strategies in their *ActiveThief*. These strategies basically evaluate all the samples in adversary's query set and collect the best *k* samples by their evaluation functions.
Uncertainty strategy is based on the *Uncertainty Sampling* by Lewis and Gale [26]. This strategy prefers samples x having higher entropy of Softmax($F_A(x)$):

$$\text{Entropy}(x) = -\sum_{i=1}^{C} p_i \log(p_i) \quad (5)$$

where C is the number of classes, and p_i is the prediction confidence softmax($F_A(x)$) at index i.

K-center strategy applies the greedy K-center algorithm by Sener and Savarese [27]. At each iteration, adversary sets the prediction probability vectors of the previously selected data as centers and cluster the rest of candidate data. Then, the adversary selects top-k distant data from their corresponding centers.
DFAL strategy perturbs every candidate data x to x' by the DeepFool-based Active Learning (DFAL) algorithm [28] at each iteration, such that $F_A(x) \neq F_A(x')$. K samples with the least perturbation $\| x - x' \|_2^2$ will be selected.
Finally, for *DFAL+K-center*, adversary first applies the *DFAL* strategy to pick *N* centers, and then use *K-center* strategy to select *k* data.

In section §IV-C, we use random strategy as a baseline to study JBAs' performance.

IV. Exploration on Jacobian-based Attacks

We notice that one of the challenges for JBA attackers is to prevent F_A from *overfitting* the query set. Starting from $1\% \sim 10\%$ budget of natural images, attackers rely on the augmentation algorithms in JBA to expand their query set. If JBA fails to add enough diversity to its query set, it almost surely leads to overfitting F_A to the query sets.

Existing JBAs ([2]–[4]) claim/assume that they are able to introduce diversity by pushing data cross or closer to F_V's decision boundaries. Intuitively, pushing data cross boundaries is similar to *adversarial example* crafting, which also changes data label, and thus encourages data to add diversity not only from its original decision region, but also its neighbor regions. Samples that are close to decision boundary are also referred to as *uncertain samples*. The claim that uncertain samples are more *informative* is true in many machine learning models. For instance, support vectors in *Support Vector Machine* are close to decision boundaries/hyper-planes and have more influence on the position and orientation of boundaries/hyper-planes than other data in the training set; at each iteration of updating the parameters of a *logistic regression*, *Iteratively Re-Weighted Least Squares* (IRLS) gives more weight to data near the decision boundary.

Generating adversarial or uncertain samples may be desirable to JBA, yet we still observe bottleneck of performance and overfitting (depending on the seed query set, accuracy $> 90\%$ and even $\sim 99\%$ on query sets, while around $65\% \sim 87\%$ on the test set) during JBA model extraction. In this section, we characterize the diversity added by JBA by the distance between data and the decision boundaries in both F_A and F_V, and compare how much JBA can improve from baselines trained on the initial query set (*i.e.* no Jacobian-based augmented data). Given that the attacker may or may not have access to the exact same dataset where F_V is trained on, we further examine how the diversity/complexity in query set impact JBA. More specifically, we present our exploration on JBAs from two angles:

1) Following the assumption/claim in prior works ([2], [4]) that JBAs work by introducing more diversity to the initial query set X_0, we take a closer look to examine

this statement and characterize the added diversity in adversarial queries from two aspects:

- Pushing queries cross the decision boundaries and thus change their labels after the Jacobian-based augmentation.
- Generating more uncertain queries, which researchers believe are more *informative*.

2) In section §IV-C, we explore how JBAs perform when the attacker chooses datasets across a spectrum of complexity relative to F_V's training set to construct $\mathcal{X}_0$.

A. Experiment Settings

1) Model Architectures: In all the experiments on JBAs and Random Sampling, we adapt the official implementation[4]. Victim uses VGG16 [29] with *batch normalization* [30] (VGG16-BN), and train the model on CIFAR10 training set to reach 93.38% accuracy on the test set[5]. We allow the adversaries to use the same model architecture as the victim (VGG16-BN), and to start from loading ImageNet [31] pre-trained weights to the model.

For DFME, we adapt the official implementation[6]. Both the victim and adversary use model Resnet34-8x [32], and the victim achieves 95.54% accuracy on CIFAR10 test set. The weights for victim is available[7].

Extraction Accuracy is evaluated on CIFAR10 test set.

2) Attack Algorithms: All the implementation are in PyTorch [33], and we use PGD [15] and CW ℓ_2 attacks implemented by Foolbox3([34], [35]). Unless specify otherwise, we use the default settings in PyTorch, Foolbox3, and other source code.

At each iteration in attack algorithms below, we choose the top-3 highest confident predictions of F_A other than victim's prediction to craft targeted adversarial examples using some augmentation algorithms.

JB-top3: We adapt the implementation and use the default settings of *I-FGSM* [2] by Orekondy *et al.*[8], and follow [36], we rename *I-FGSM* to *JB-top3*.

JBA-PGD and **JBA-CW_l2:** As discussed in section §III-A, Algorithm 2 can be any algorithm for crafting adversarial examples. In JBA-PGD we use targeted ℓ_∞ PGD with $\epsilon = 8/256$ with all the default settings, and in JBA-CW_l2, we use targeted CW ℓ_2 attack [16], with 150 steps and 5 binary search steps.

JBA-BS: We set $\epsilon = 1e-6$ and $N_{\text{max}} = 1000$

DFME: We follow all the defualt setting in the official implementation.

Random Sampling (RS): As discussed in section §III-C, we compare JBAs against the random sampling strategy in Active Learning Extraction Attacks using different query sets.

[4]https://github.com/tribhuvanesh/prediction-poisoning/tree/master/defenses/adversary

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

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

[7]https://github.com/VainF/Data-Free-Adversarial-Distillation#0-download-pretrained-models-optional

[8]https://github.com/tribhuvanesh/prediction-poisoning/blob/master/defenses/adversary/jacobian.py

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

3) Query Sets: Since the victim train F_V on CIFAR10, we randomly sample a subset of CIFAR10 training set to be $\mathcal{D}_0$. Our threat model allows adversaries to use other publicly accessible datasets, so we additionally choose 8 other datasets ([31], [37]–[43]) and use their training set as query sets so that adversaries can either sample seed images or queries from these datasets. However, in our experiments, adversaries do not sample queries from any union of datasets.

B. Diversity in Adversarial Queries

In [4] and [2], researchers claim/assume that JBAs work by generating diverse samples exploring decision regions and boundaries in DNNs. Indeed, we observe that the similarity encoder in our detector (more details in section §VI-A) extracts diverse feature factors from adversarial queries. To further characterize and quantify how much diversity added by JBAs, we measure the number of adversarial queries crossing decision boundaries and going to other decision regions in F_A and F_V, after being Jacobian-based augmentation.

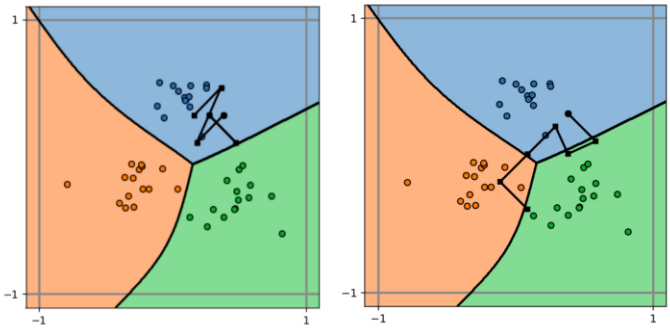

Fig. 1. Illustrations we borrow from [2]: decision regions are colored differently and separated by black curves; black dots are adversarial queries, and the lines connecting them trace their positions as augmentation iterates. **Left**: We observe that JBAs do introduce additional features to $\mathcal{D}_0$. **Right**: We are curious about whether pushing queries cross decision boundaries helps improve extraction accuracy, and how many of them cross the boundaries in F_A and F_V.

1) Comparing to baselines: To answer these questions, we first build a baseline to see how much JBAs improve extraction accuracy from that. We give all JBAs $50,000$ query budget. For JBAs collecting $\mathcal{X}_0$ with size N_{seed}, the baseline is RS with N_{seed} budget. Table I and table II summarize the experiment results.

In table III, we record the extraction accuracy by DFME when the attacker sends 50K, 500K, 5M, and 20M queries because we find it helpful to shed some light on the characteristics of adversarial queries. Due to initialize weights randomly, the baseline for DFME is simply a random guess: given that CIFAR10 has 10 classes, that will be 10%. And our experiment show that after sending 50K queries, the extraction accuracy is still 10%. However, being granted enough amount of budget, DFME can obtain 88% extraction accuracy.

TABLE I
EX. ACC. ON CIFAR10 ($N_{\text{SEED}} = 500$)

Scheme	Ex. Acc.
Baseline	66%
JB-top3	67%
JBA-PGD	72%
JBA-CW_ℓ_2	61%
JBA-BinarySearch	63%

TABLE II
EX. ACC. ON CIFAR10 ($N_{\text{SEED}} = 5000$)

Scheme	Ex. Acc.
Baseline	85%
JB-top3	87%
JBA-PGD	87%
JBA-CW_ℓ_2	85%
JBA-BinarySearch	85%

Takeaway: Comparing table I and table II, the marginal effect on how much JBAs improve from the baselines is apparent: from up to 6% improvement when $N_{\text{seed}} = 500$ to up to 2% when $N_{\text{seed}} = 5,000$. Our experiments imply that there is an upper bound on the improvement brought by Jacobian-based augmentation. Nonetheless, JBAs are useful when the attacker can only have access to less then 1% of F_V's training data. Given that an extraction accuracy of 87% is close to the oracle accuracy 93%, we could possibly attribute this marginal effect to model capacity.

Interestingly, this marginal effect does not show up in DFME. Although using $400\times$ more budget, DFME is able to converge to even slightly better extraction accuracy then JBAs, and have 78% improvement. This implies that the upper bound of improvement in JBAs is not due to something like *the curse of dimensionality*, but their own limitations. Figuring out what helps DFME escape from the marginal effect in improvement can shed some light on the future direction of designing more effective augmentation algorithms for JBAs. In section §IV-B3, we will compare the amount of uncertainty samples in DFME and JBAs.

2) Diversity by Crossed Samples: To see how many adversarial queries cross decision boundaries and go to other decision regions after augmentation, we trace the proportion of *crossed samples* at each iteration: we define data x is a crossed sample if $L_F(x)$, the prediction label of a model F on x, change after augmentation by *Aug*:

$$L_F(x) \neq L_F(x') \text{ where } x' = Aug(x, F)$$

And we plot the result in figure 2 for $N_{\text{seed}} = 500$, and in figure 3 for $N_{\text{seed}} = 5,000$.

Since we generate three new samples from each one of the sample in the previous query set, JBAs applying $N_{\text{seed}} = 500$ use up their budget in 4 iterations, leaving us 4 observation in figure 2. While for $N_{\text{seed}} = 5,000$, budget is exhausted in 2 iterations, leaving us too few data points to see any tendency. In order to increase the number of iterations to see the tendency more clearer with $N_{\text{seed}} = 5,000$, we randomly sample 400 data to augment from $\mathcal{D}_T$ at each iteration(also referred as *reservoir sampling* in [2], [4]). Hence, there are $\lceil \frac{45,000}{12} \rceil = 38$ iterations in total. We find that whether use *reservoir sampling* or not does not change our conclusion.

TABLE III
EX. ACC. OF DFME ON CIFAR10

# Queries	50K	500K	5M	20M
Ex. Acc.	10%	14%	71%	88%

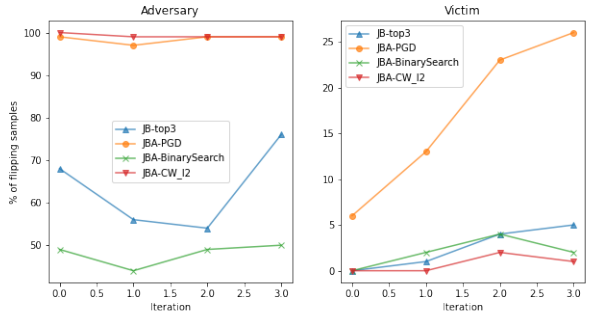

Fig. 2. $N_{\text{seed}} = 500$: Percentage of crossed samples at each iteration. Left: crossed samples for F_A. Right: crossed samples for F_V

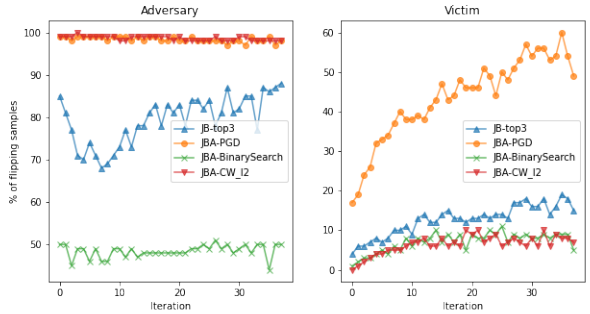

Fig. 3. $N_{\text{seed}} = 5,000$: Percentage of crossed samples at each iteration. Left: crossed samples for F_A. Right: crossed samples for F_V

So does adding diversity by pushing samples cross the boundaries help? Yes, to some extent. Table I and figure 2 tell us that the attack starting with 1% of the F_V's training set $\mathcal{X}_V$, JBA-PGD has the highest extraction accuracy 72%, improving 6% from the baseline, and its new synthesized data at each iteration also has the highest proportion of cross samples for F_V comparing to other JBAs. However, for JBA-CW_ℓ_2, almost 100% of the generated data are crossed samples for F_A, but less than 5% of them are crossed samples for F_V. The extraction accuracy of JBA-CW_ℓ_2 is lower than the baseline due to overfitting. We know it is overfitting since JBA-CW_ℓ_2's training accuracy on the final synthesized data D_T (containing 50K data) is 98%, while JBA-PGD is 92% – overfitting as well, but less severe.

Nevertheless, table II and figure 3 tell us that if the attacker has a larger seed query set (10% of $\mathcal{X}_V$), the best improvement from the baseline is only 2%, even though JBAs can generate more crossed samples for F_V.

3) Diversity by Uncertain Samples: In addition to pushing samples to different decision regions, some researchers believe that near-boundary queries bring more uncertainty and thus are more informative for model extraction attacks, so researchers apply uncertainty sampling to augment $\mathcal{D}_0$. Researchers often use the *predicted confidence* (top-1 highest confident class) given by a classifier F on a input x as a proxy to measure the distance from x to its nearest decision boundary in F. Based on the setting above, Yu *et al.* [3] claim that their *Feature Fool* (FF) attacks can "efficiently learn the distance between decision boundaries of the victim model and the stolen model." Although we do not implement FF, it suffices to study JBA-CW_ℓ_2 because:

- The experiments in [3] show that in terms of extraction accuracy, JBA-CW_ℓ_2 performs similarly to FF, and in some cases even better;
- We verify that CW_ℓ_2 can also generate uncertain samples;
- Both FF and CW_ℓ_2 only have access to F_A, but not F_V.

We want to take a closer examination to see 1) whether JBAs can generate uncertain samples for F_V, and 2) whether those uncertain queries generated by JBAs help improve extraction accuracy.

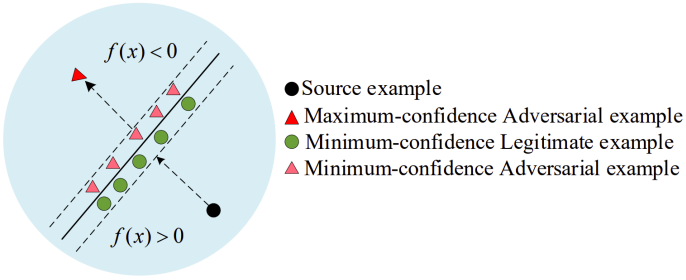

Fig. 4. An illustration we borrow from [3]: researchers believe near-boundary queries are helpful to model extraction attacks

We plot the histogram of *predicted confidence* given by both F_A and F_V on adversarial queries in figure 5. We find that the histograms for $N_{\text{seed}} = 500$ and $N_{\text{seed}} = 5,000$ are close enough that will not affect our conclusion, so we choose the one for $N_{\text{seed}} = 5,000$.

First, compared the predicted confidence histograms of F_A among 4 Jacobian-based attacks, the ones by JBA-CW_ℓ_2 and JBA-BinarySearch verify that CW_ℓ_2 and our binary search attack are effective at generating uncertain samples for F_A. As a side-note, *uncertainty* could also implies low-distortion. For instance, CW_ℓ_2 produces $1 <\| x - x' \|_2< 2.5$, while PGD produces $5 <\| x - x' \|_2< 6$. However, the highly left-skew histograms of F_V by these four attacks in figure 5 also suggest that F_V remain high-confident to most of the generated queries, which implies the poor transferability of these uncertainty samples: they remain far away from decision boundaries of F_V no matter how they locate in F_A. Furthermore, poor

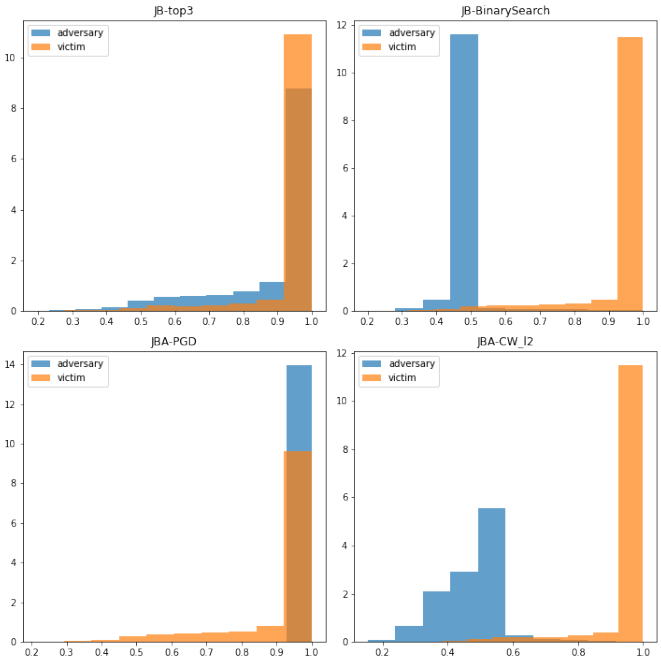

Fig. 5. JBAs: Histogram of predicted confidence

transferability implies low similarity between the boundaries of F_V and F_A. Uncertain samples only for F_A cannot help the attacker train F_A's boundary to become similar to F_V's.

Combining figure 3 and figure 5, we may shed some light on the characteristics of adversarial queries generated by different JBAs: JB-top3 and JBA-PGD push samples cross decision boundaries and even *overshoot* them too far away from the boundaries. This makes sense since they both do not have regulation term relevant to predicted confidence in their objective function; JB-CW_ℓ_2 and JB-BinarySearch are able to generate uncertain samples for F_A, but they have poor transferability to F_V, and the majority of them are not crossed samples to F_V, either.

In contrast to JBAs, we find that DFME can generate uncertain samples for both F_V and F_A even when the number of queries is 50K and extraction accuracy is only 10% at that moment. Furthermore, we can also observe that the predicted confidence distribute much more uniformly from 0 to 1 than in JBAs, and distributions of F_A and F_V start synchronizing at least after sending 5M queries. Based on these observations, we have the conjecture that an effective MEA need to be able to generate queries not just through pushing samples cross decision boundaries, or generate uncertain samples, but thoroughly exploring the whole input space.

C. Impacts of Query Set Selection

We measure the effectiveness of JB-top3 and JBA-PGD under different threat models. An attacker may not have access to any of the samples from the exact same training set used to train the victim model. Instead, a real-world attacker may have access to some seed samples from a different distribution.

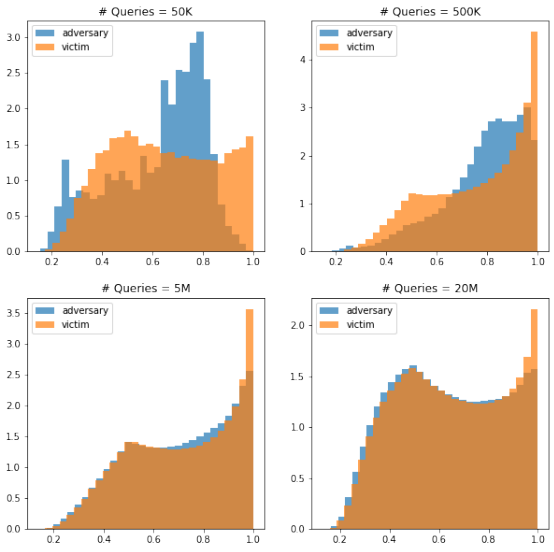

Fig. 6. DFME: Histogram of prediction confidence with different number of queries

We use 8 public datasets as different types of seed samples for the attackers.

TABLE IV
EX. ACC. ON CIFAR10: RS V.S. JBAS ($N_{\text{SEED}=500}$)

QuerySet (Size) Budget: # Seed Images:	**RS** ***500*** (*Baseline*)	**JB-top3** 50,000 ***500***	**JBA-PGD** 50,000 ***500***
CIFAR10 [38]	66%	67%	**72%**
TinyImageNet [37]	46%	57%	**61%**
ImageNet1k [31]	44%	57%	**62%**
CIFAR100 [38]	43%	55%	**58%**
SVHN [39]	19%	24%	**32%**
CINIC10 [40]	58%	64%	**69%**
Indoor67 [41]	33%	44%	**51%**
CUBS200 [42]	20%	23%	**31%**
Caltech256 [43]	44%	49%	**58%**

Following the previous section (section §IV-B1), we examine the impacts of query set selection by comparing JB-top3 and JBA-PGD against their RS baselines. JBAs have $50,000$ query budget. For JBAs collecting $\mathcal{X}_0$ with size N_{seed} from a dataset $\mathcal{D}$, the baseline is RS with N_{seed} budget from the same dataset.

Table V and table IV summarize the comparison, and as a reference, we include the extraction accuracy of RS using the 50K budget in table VI. Note that Indoor67, CUBS200, and Caltech256 (have 14,280, 6,033, and 23,380 images in their training sets, respectively) have less than 50K data in their training set, and thus RS in table VI using 50K budget will not be able to consume all budget. For these three datasets, RS simply terminates after it exhausts all the data in the query sets, while JBAs are able to use up all 50K budgets by Jacobian-based augmentation.

TABLE V
EX. ACC. ON CIFAR10: RS V.S. JBAS ($N_{\text{SEED}=5,000}$)

QuerySet (Size) Budget: # Seed Images:	**RS** ***5,000*** (*Baseline*)	**JB-top3** 50,000 ***5,000***	**JBA-PGD** 50,000 ***5,000***
CIFAR10 [38]	85%	**87%**	**87%**
TinyImageNet [37]	69%	**78%**	**78%**
ImageNet1k [31]	67%	**78%**	**78%**
CIFAR100 [38]	68%	**78%**	**78%**
SVHN [39]	24%	40%	**43%**
CINIC10 [40]	78%	**84%**	**84%**
Indoor67 [41]	50%	70%	**71%**
CUBS200 [42]	31%	46%	**54%**
Caltech256 [43]	63%	**77%**	**77%**

TABLE VI
EX. ACC. ON CIFAR10: RANDOM SAMPLING

QuerySet (Size) Budget:	**RS** *50,000*
TinyImageNet [37]	84%
ImageNet1k [31]	85%
CIFAR100 [38]	84%
SVHN [39]	36%
CINIC10 [40]	89%
Indoor67 [41]	63%
CUBS200 [42]	36%
Caltech256 [43]	78%

In general, JBAs can extract a model with higher accuracy compared to the baseline RS adversary. The amount of accuracy improvement varies from 2% to 21%, depending on the where seed images come from and how many of them are available to the attacker. Our experiments suggest that when adversary has limited amount of data as the $\mathcal{X}_0$, JBAs outperform a RS attacker.

Furthermore, we can conclude that high quality data for MEA do not necessarily come from those large and universal datasets: if we rank the extraction accuracy from low to high (figure 7 and figure 8), we can see that as the dataset getting larger and more complicated, the baseline and resulting extraction accuracy getting higher (SVHN $\rightarrow$ CIFAR100); to some point, complexity in dataset does not help improve the extraction accuracy, but the similarity to F_V's training set (ImageNet1k $\rightarrow$ CIFAR10) does. Table VI tells us that even when the attacker can collect large amount of data, JBAs still outperform RS attack if the data quality are *low* (SVHN, Indoor67, and CUBS200), but when the attacker has access to decent quality data, RS attack is a better option.

Takeaway:

- JBA has an advantage over other MEA methods when the attacker can only collect limited amount of data, or when the initial query set has too little overlap with victim's task domain;
- The quality of MEA queries is positively correlated to the similarity between the query set and victim's training set.

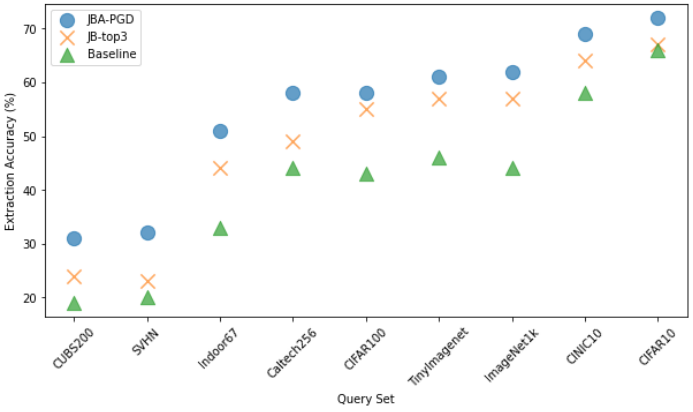

Fig. 7. $N_{\text{seed}} = 500$: extraction accuracy ranked by X_0

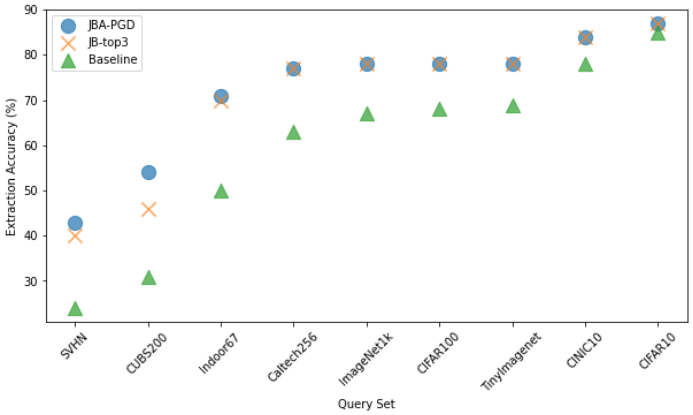

Fig. 8. $N_{\text{seed}} = 5,000$: extraction accuracy ranked by X_0

V. Existing Defenses

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

A. PRADA

Juuti *et al.* [2] propose their detector PRADA based on the observation that the ℓ_2 distance among benign queries follows normal distribution, and their detector monitors the distribution of $\min_{x \in \text{history}} \| x - x_{\text{new query}} \|_2$ and check the *normality* of this distribution by running Shapiro-Wilk normality tests.

Unfortunately, ℓ_2-norm is not stable and many prior works ([3], [5]) have demonstrated that attacker can bypass PRADA. Chen *et al.* show that using feature vectors extracted by similarity encoder is a more robust query representation for detection.

B. Out-of-distribution Detector

Atli *et al.* [44] and Kariyappa *et al.* [45] propose detector based on out-of-distribution (OOD) detection. Atli *et al.* train a binary classifier to tell whether a query comes from the same distribution as victim's training set. Kariyappa *et al.* fine-tune F_V such that it tends to output high top-1 prediction confidence ($> 90\%$) on *in-distribution* data, while the top-1 prediction confidence for OOD data distribute uniformly from 0 to 1. Essentially, this type of detector assume all the OOD data are adversarial, while in our work, we assume all the natural data are benign.

C. Prediction Poisoning/Modification

Orekondy *et al.* [36] propose prediction poisoning that perturbs victim's output prediction probability vector. More specifically, given a loss function L, they add perturbation ϵ^* to a output y such that

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

while maintaining API utility. In other words, ϵ^* is a misinformation that give a wrong direction to adversaries when they use this contaminated output to train F_A on some loss L. But attacker can easily post-process the output into a one-hot vector (like in our setting) to avoid this poison output.

Kariyappa *et al.* [46] show that discontinuous predictions on adversarial queries prevent the attacker from effectively stealing models. In their threat model, any data coming from different distribution than the victim's training set are consider malicious. They propose *Ensemble of Diverse Models* (EDM) that is able to produce inconsistent predictions on out-of-distribution (OOD) data. More specifically, the defender trains a collection of F_Vs producing accurate prediction on in-distribution data, while F_Vs produce inconsistent predictions on OOD data. Incoming queries go through a secret perceptual hashing they propose, and is assigned to one of the victim's models for prediction according to the hashing.

D. Detectors against Black-box Attacks

Similar to MEA, black-box adversarial example attacks (AEAs) ([47]–[50]) send a sequence of queries to F_V in order to craft adversarial examples. We show that defenses for black-box AEAs can inspire the design of detection scheme for MEA. Our similarity-based detector is inspired by the *Stateful Detector* (SD) for AEAs proposed by Chen *et al.* [5] (more details in section §VI). SD extracts queries' feature vectors and store it in its history. It then monitors the a query's average distance to its *k*-nearest neighbors, and terminates the account once this distance is below some threshold.

Blacklight by Li *et al.* [6] is another detector for black-box AEAs. Blacklight also exploits the similarity among adversarial queries, hence it is possible to build a MEA detector based on *Blacklight*. It computes a set of *fingerprints* for each query through secure on-way hashes, and it searches for similar queries by hash table lookup.

VI. Our Detectors

In this section, we introduce two detectors to protect the model against learning-based attacks, specifically Jacobian-based augmentation attacks (JBA) (algorithm 1) and Data-free attacks (DFA) (introduced in section §III-B). They are the similarity-based detector in section §VI-A and the VAE-based detector in section §VI-B.

In particular, our detector functions at step 5 in the Algorithm 1, keeping an eye on the received queries and terminating users' accounts if any suspicious behavior is detected.

A. Similarity-based Detector

This method is inspired by the *Stateful Detector* against black-box adversarial examples crafting by Chen *et al.* [5]. While their detector monitors if the average distance of a query to its k-nearest neighbors below some threshold, we empirically find that the number of similar pairs is a better indicator to malicious behaviors. A similar pairs (q_i, q_j) is a pair of queries that the ℓ_2 distance between their feature vectors $f(q_i)$ and $f(q_i)$, respectively, is less than some threshold δ:

$$(q_i, q_j)_{\text{similar}} \coloneqq (q_i, q_j) \text{ such that } \ell_2(f(q_i), f(q_j)) < \delta$$

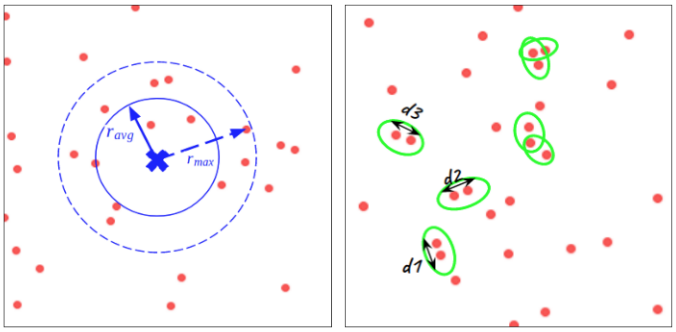

Fig. 9. Illustration of detection scheme when queries feature vectors are embedded in 2-dimensional space
Left: Detection scheme in [5], where r_{avg} is the average ℓ_2-distance to the k-nearest queries; defenders expect r_{avg} to be less than some threshold when adversaries are sending malicious queries. **Right:** Our scheme counts the number of similar pairs (circled in green), where the ℓ_2-distance of their feature vectors is less than some threshold.

We train a similarity encoder with the same architecture as in [5] to extract the feature vectors of queries (more details in section §VIII-A). In general, an encoder f first train on a classification problem and learn to extract features, then we train the encoder on the same dataset to minimize the contrastive loss function [51]:

$$\begin{aligned} \min \mathcal{L}(x_0, x_+, x_-, f) = &\| f(x_0) - f(x_+) \|_2^2 \\ &+ \max\{0, m^2 - \| f(x_0) - f(x_-)\} \end{aligned} \tag{6}$$

where x_0 is a natural image, x_+ is x_0 after slight transformation, and x_- is a different natural image in the dataset. We find that $m = \sqrt{10}$ suffices to train a good encoder in our settings, and we tune the threshold δ such that the FPR on the tuning set is 0.01%.

B. VAE-based Detector

In this section, we will first briefly introduce Variational Autoencoder (VAE), and then we will introduce how we take advantage of VAE to build our detector.

VAE is a type of generative model consisting of a encoder and a decoder. Encoder E encodes input x as a distribution $q_x(z)$ over the latent space; to generate data, VAE samples z from $q_x(z)$ and decoder D decodes z as the generated data:

$$x \xrightarrow{\text{encoder}} q_x(z) \xrightarrow{\text{sample } z \sim q_x(z)} z \xrightarrow{\text{decoder}} D(z) = x'$$

The likelihood of x' evaluates how close x' is to x. If we assume the reconstruction error $x - x' = \delta \sim \text{Gaussian}(0, \sigma)$, then the likelihood of x' can be written as:

$$\text{Likelihood}(x'|x, E, D) = \frac{1}{\sqrt{2\pi\sigma}} \exp\{-\frac{1}{2}\left(\frac{x - x'}{\sigma}\right)^2\}$$

Further, we take the natural log of this likelihood and get:

$$\log p(x'|z) = l(x'|x, E, D) \sim -\left(\frac{x - x'}{\sigma}\right)^2$$

If x is an image, then the log-likelihood of the reconstructed image x' will be proportional to the negative mean square error (MSE) between x and x'. Indeed, many VAEs use $\text{MSE}(x, x')$ as their reconstruction loss during the training.

An adversary using JBA or DFA must perturb or generate their queries with features that E and D never saw during their training time. Therefore, we expect the log-likelihood for an adversarial query is lower than for a benign one. To add more flexibility to the detector and allow us to tune its FPR, we create a *credibility interval* for the estimation of log-likelihood. The credibility interval includes values that are within $\alpha\times$ standard deviation away from an estimation. We tune $\alpha > 0$ such that the FPR on tuning set is 0.01%. The detector monitors the average of estimated log-likelihood and calculate the sample standard deviation of this value as F_V is receiving queries. Detector considers a query malicious if the upper bound of the estimation is lower than the average log-likelihood of the benign users.

Besides checking if the upper bound lower than the threshold, one can possibly additionally check if the lower bound is higher than the threshold. However, we choose not to do so because we often observe a scenario where the lower bound higher than the threshold when adversary sending first several natural images, as illustrated in figure 10. This makes our tuning for FPR tricky, since some detected queries are natural images, which should be consider benign by the design of our threat model. There may be ways to design a better credibility interval to avoid this problem, but we do not explore this in this paper.

There are many powerful VAEs in existing works, and here we choose VQ-VAE proposed by Oord *et al.* [52] to build our VAE-based detector. The latent space of VQ-VAE is discrete in $z \in \{1, 2, \ldots, K\}$ with corresponding embeddings $e_1, e_2, \ldots, e_K$. Encoder encodes and maps each pixel in the input image to one of the latent space embeddings, and then decoder reconstructs the image from this discretely embedded version of input.

It is possible to use ELBO, instead of $\log p(x'|z)$, to estimate the benign likelihood. ELBO bounds $\log p(x)$, and particularly for VQ-VAE:

$$\log p(x) \geq \log p(x'|z)p(z)$$

where $p(z)$ is the prior for the sampled latent variable. However, we do not do so, since 1) the prior distribution is independent from the training of VQ-VAE: Oord *et al.* trains a PixelCNN [53] to *learn* the prior distribution over z in an autoregressive

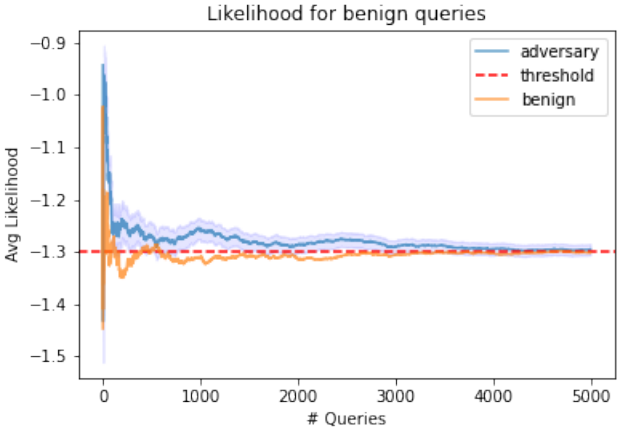

Fig. 10. Illustration of the scenario where the lower bound of average log-likelihood estimation higher than the threshold when the adversary is setting benign images.

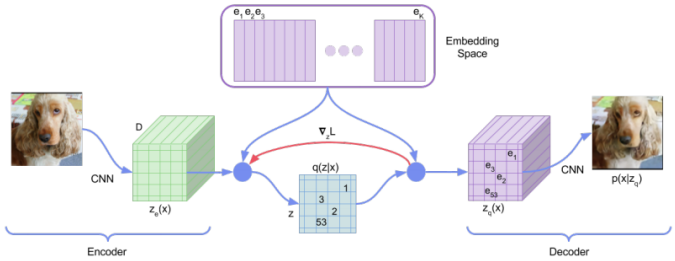

Fig. 11. An illustration we borrow from [52]. The encoder maps pixels into discrete latent space, and converts this encoded input to the decoder's input through an embedding lookup.

way once the training of VQ-VAE is done; 2) we find it difficult to fit a satisfying prior distribution that is beneficial to our detector. Nonetheless, we believe using other VAE architectures and ELBO could possibly yield a better detector.

VII. Adaptive Attacks

An effective detector should be robust enough to work even after the attacker is aware of its deployment and understand how it works. In this section, we introduce two adaptive attacks that are specifically designed against our detectors: *query blinding* in section §VII-A and *query filtering* in section §VII-B. In general, a JBA attacker deploys an adaptive attack at step 5 in the Algorithm 1, before sending the queries to F_V, which is potentially protected by a defender.

We also implement our adaptive attacks for DFME attackers [20]. Recall from section §III-B that in one training epoch, 1) DFME first trains the generator G, then 2) uses G to generate data and ask F_V to label them for F_A's training. Both of these steps need to send queries to F_V, but without the gradient approximation in the first step, DFME will not effectively work. Hence, we deploy our adaptive attacks in the second step, before sending queries to F_V.

A. Query Blinding

Query blinding (QB), proposed by Chen *et al.* [5], is a general strategy against detector exploiting any forms of similarity among adversarial queries. QB hides an actual query x by some random transformation function t, which has a parameter r to control the degree of randomness. Then, adversary send $x' = t(x; r)$ to F_V, and the goal is to obtain a output that is similar to $F_V(x)$ without revealing x to the detector. With QB, step 5 becomes $\mathcal{Y}_T \leftarrow F_V(t(\mathcal{X}_T; r))$ and the rest of Algorithm 1 remains unchanged, including the step 6.

Fig. 12. **Left:** Original input images; **Right:** Output images of a converged auto-encoder blinding function: the auto-encoder learns to allocate the input random noise to positions that have minimal impact on model outputs.

Following [5], we choose a wide range of random image transformations (*e.g.* rotation, scaling, random noise, etc.) to test our detector apply their corresponding default r in [5]. We also follow the setting and architecture in [5] to examine an auto-encoder blinding function. An auto-encoder blinding function t_{auto} takes in a random noise $z \sim \mathcal{N}(0, 0.095)$ and the original image x, then we train t_{auto} such that for a model F:

$$F(t_{\text{auto}}(x, r)) \approx F(x)$$

As illustrated in figure 12, a converged auto-encoder blinding function can re-distribute the input noise so that it has minimal impact on model outputs. Given our threat model, the attacker train this the auto-encoder and F on either the subset of F_V's training set $\{(\mathcal{X}, \mathcal{Y})\}_V$, or a publicly accessible dataset that is the most similar to $\{(\mathcal{X}, \mathcal{Y})\}_V$.

B. Query Filtering

In this work, we propose *query filtering*, another general strategy against MEA detectors, specifically to evade stateful detector. The goal of QF is to allow the attacker to locally emulates a stateful detector and remove those queries that can possibly raise detector's alarm. For instance, if the attacker realizes that the defender is using similarity-based detector to protect F_V, then the attacker cam implement QF by training a substitute similarity encoder E_{sub} and tune a substitute detection threshold δ_{sub} to build a substitute detector D_{sub}. The attacker thus can remove those queries which D_{sub} believes are suspicious. Again, due to our threat model, the attacker can train this D_{sub} on either the subset of F_V's training set $\{(\mathcal{X}, \mathcal{Y})\}_V$, or a publicly accessible dataset that is the most similar to $\{(\mathcal{X}, \mathcal{Y})\}_V$.

VIII. Evaluations on Detectors

Evaluation settings follow section §IV-A unless state otherwise. JBAs sample 5,000 seed images and has 50K budget. All the accuracy scores (in %) are rounded to their nearest integers, and vary within $\pm 1\%$. Examination of similarity-based detector is presented in section §VIII-A; and the results on VAE-based detector is in section §VIII-B.

A. Similarity-based Detector

As introduced in section §VI-A, our similarity-based detector monitors the similarity of the incoming queries to queries in the history. Defenders can terminate the users' accounts once they find too many similar pairs.

1) Settings: We use the same architecture and method in [5] to train our similarity encoder and query blinding autoencoder. In our setting, an account is terminated if there are 50 similar pairs. We tune the detector threshold=0.12986328125 by binary search on both the training and testing sets of the CIFAR10 and CINIC10, so that the false positive rate (FPR) reaches 0.01%. We use the default parameter r in [5] for our query blinding adaptive attack.

2) Effectiveness: In table VII, JBAs construct $N_{\text{seed}} = 5,000$ $\mathcal{X}_0$ from CIFAR10 and use 50K budgets. The table suggests that JB-top3 and JBA-PGD use least amount of accounts to mount MEA.

TABLE VII
Similarity-based Detector against JBAs

JBA	# Accounts	Ex. Acc. on CIFAR10
JB-top3	59	87%
JBA-PGD	**21**	87%
JBA-CW_ℓ_2	126	85%
JBA-BinarySearch	765	85%

Table VIII reports the number of accounts needed when the attacker collect $\mathcal{X}_0$ from different datasets. In table IX, we report the false positive rate (FPR) calculated under the benign usage of that dataset (*i.e.* by sending ALL natural images in the dataset). Note that since we tune the threshold on both CIFAR10 and CINIC10 so that our detector has FPR 0.01% on the concatenation of them, but table IX evaluate the FPR on each dataset separately, so the FPR on CIFAR10 and CINIC10 are not exactly 0.01%.

TABLE VIII
Similarity-based Detector for JB-top3 and JBA-PGD

	JB-top3		JBA-PGD	
Query Set	# Accounts	Ex. Acc.	# Accounts	Ex. Acc.
CIFAR10	59	87%	21	87%
TinyImageNet	96	78%	25	78%
ImageNet1k	90	78%	24	78%
CIFAR100	101	78%	26	78%
SVHN	221	40%	43	43%
CINIC10	54	84%	18	84%
Indoor67	90	70%	25	71%
CUBS200	94	46%	24	54%
Caltech256	73	77%	16	77%

TABLE IX
FPR of Similarity-based Detector

Query Set	FPR
CIFAR10	0.008%
TinyImageNet	0.008%
ImageNet1k	0.011%
CIFAR100	0.007%
SVHN	0.060%
CINIC10	0.011%
Indoor67	0.013%
CUBS200	0.008%
Caltech256	0.007%

From table VIII, we can see how effective our similarity-based detector is on different threat model. In general, JBA-PGD uses less amount of accounts than JB-top3. For attackers having access to some victim's training data $\mathcal{X}_V$ (in our case is CIFAR10), it takes JB-top3 59, and JBA-PGD 21, accounts to extract a surrogate model with 87% accuracy on victim's task. For other attackers who do not have access to $\mathcal{X}_V$, it generally takes them more accounts to extract a surrogate model (except for CINIC10 and Caltech256).

Combining table VIII and table IX, one can find that although the similarity encoder is trained on CIFAR10, and the threshold is only tuned on CIFAR10 and CINIC10, our similarity-based detector is able to detect and terminate many adversarial accounts while keeping FPR $0.01 \pm 0.003\%$, regardless of the sources of $\mathcal{X}_0$. This suggests that similarity-based detector can be a powerful yet easily-deployed detection scheme against Jacobian-based extraction attacks.

3) Adaptive Attack: We show that our similarity-based detector is robust against adaptive attacks. As discussed in section §VI-A, adaptive attacks include *query blinding* (QB, section §VII-A) and *query filtering* (QF, section §VII-B). In table X and table XI, we summarize the results of testing the similarity-based detector against adaptive JB-top3 and JBA-PGD that construct $\mathcal{X}_0$ from CIFAR10. Table XII presents the results when JB-top3 attacker construct $\mathcal{X}_0$ from CINIC10. Table XIII presents the results on vanilla and adaptive DFME attack.

TABLE X
ADAPTIVE JB-TOP3 WITH CIFAR10 $\mathcal{X}_0$

Adaptive Scheme	**Query Set**	**# Accounts**	**Ex. Acc.**
Non-adaptive	**CIFAR10**	59	87%
Query Filtering	Trained on *full cinic10*	55	82%
	Trained on *cifar10 seed*	11	86%
Query Blinding	Crop	50	85%
	Brightness	221	77%
	Scale	28	86%
	Rotate	59	86%
	Contrast	266	79%
	Uniform	245	79%
	Gaussian	179	77%
	Translate	19	85%
	Auto-encoder	6	81%

TABLE XI
ADAPTIVE JBA-PGD WITH CIFAR10 $\mathcal{X}_0$

Adaptive Scheme	**Query Set**	**# Accounts**	**Ex. Acc.**
Non-adaptive	**CIFAR10**	21	87%
Query Filtering	Trained on *full cinic10*	21	87%
	Trained on *cifar10 seed*	8	87%
Query Blinding	Crop	26	86%
	Brightness	88	83%
	Scale	16	86%
	Rotate	18	87%
	Contrast	226	82%
	Uniform	91	82%
	Gaussian	84	82%
	Translate	13	86%
	Auto-encoder	2	82%

TABLE XII
ADAPTIVE JB-TOP3 WITH CINIC10 $\mathcal{X}_0$

Adaptive Scheme	**Query Set**	**# Accounts**	**Ex. Acc.**
Non-adaptive	**CINIC10**	54	84%
Query Filtering	Trained on *full cinic10*	58	83%
	Trained on *cifar10 seed*	8	82%
Query Blinding	Crop	58	82%
	Brightness	228	70%
	Scale	28	82%
	Rotate	58	83%
	Contrast	320	70%
	Uniform	244	69%
	Gaussian	186	68%
	Translate	26	82%
	Auto-encoder	4	67%

TABLE XIII
SIMILARITY DETECTOR AGAINST DFME (20M QUERIES)

Adaptive Scheme	**Data-free**	**# Accounts**	**Ex. Acc.**
Non-adaptive		56981	89%
Query Filtering	Trained on *full cinic10*	126674	87%
Query Blinding	Uniform	57394	87%
	Gaussian	57222	88%
	Scale	56981	88%
	Brightness	56994	89%
	Rotate	56981	89%
	Contrast	112231	70%
	Crop	56981	88%
	Translate	56981	85%

Among all the adaptive attacks, QF trained on CIFAR10 seed images performs the best in table X, table XI, and table XII, while QF trained on the full CINIC10 does not effectively bypass the detector. This suggests that when the attacker has access to part of F_V's training set data $\mathcal{X}_V$, our QF can reduce the effectiveness of similarity-based detector, but when the attacker does not have this advantage, even a substitute similarity encoder trained on data that distribute similarly to $\mathcal{X}_V$ does not help filtering out suspecious data. Among all the QBs, random scaling works the best: reduce the number of accounts while not losing extraction accuracy too much. Auto-encoder QB can reduce the number of account even more, but the resulting extraction accuracy also drops severely.

With table XIII, we show that our similarity-based detector can also work with data generated by GAN and detect DFME attack. Comparing to JBA attacker, DFME attacker needs $400\times$ more queries and $\sim 965\times$ more accounts to extract a copy of surrogate model that perform slightly better (2%) than JBA's. As a reminder that strictly speaking, DFME is not supposed to use any natural images and requires F_V to return soft-labels. But as discussed in section §III-B and section §II-B, for evaluation purpose, we allow DFME attacker to use the CINIC10 and relax our threat model to allow DFME to see logit scores, rather the hard-labels. As explained in section §VII, we deploy adaptive attacks before $G(z)$ is sent to F_V for labeling and leave the gradient approximation part untouched. Table XIII also shows that QF and QB are not effective at bypassing detector. Note that QF-CINIC10 roughly double the number of accounts: QF filters out and rejects to send $\sim 80\%$ of the generative queries, and in order to use up all 20M budget, the number of iterations is roughly double.

B. VAE-based Detector

1) Likelihood changes: As introduced in section §VI-B, our likelihood-based detector monitors the average log-likelihood while victim's model receiving queries. Figure 13 plots the change in queries' average log-likelihood before (left) and after (right) adversaries send queries augmented by Jb-top3. The orange line is the change by sending 50K natural images in CIFAR10 training set, while the blue line represents the change by Jb-top3 with 5000 seed images and 50K as its total budget. We can see that in the first 5000 queries, two lines converge to the same value, but once the adversary starts sending augmented images, its log-likelihood declines.

2) Settings: We use VQ-VAE proposed by Oord *et al.* [52] to calculate the likelihood of a query. As discussed in section §VI-B, the reconstruction loss, which is the mean-square-error, between the original image and the reconstructed image is a proxy of the negative log-likelihood of the original image. We train our VQ-VAE on CIFAR10 training set using this implementation[9] and the default setting reported in [52].

[9]https://github.com/zalandoresearch/pytorch-vq-vae

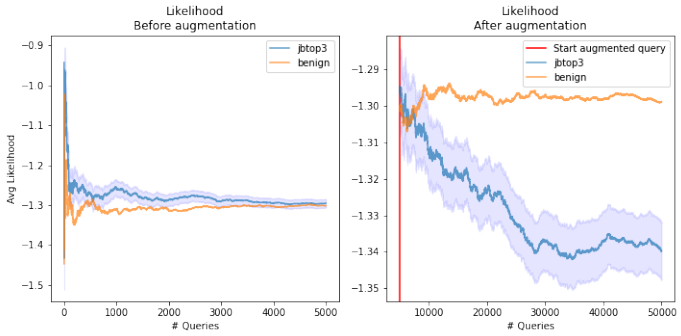

Fig. 13. Change in average log-likelihood over the number of queries.

One may notice that the estimate of log-likelihood fluctuates a lot in the first few queries. Therefore, we allow the detector *warms up* in the first 50 queries to stabilize its log-likelihood estimation. In other words, our detector starts comparing against threshold after 50 queries.

In order to tune the sensitivity of this detector, we create a credibility interval around the estimated log-likelihood (the grey area in figure 13). Only when the upper bound of the likelihood is lower than the some *threshold*=α, does the defender cancel users' accounts due to potential malicious usages. Ideally, the threshold here is the average log-likelihood of all the natural images in the problem-domain of victim's model. Values within an interval is within β-times standard deviation around the estimated log-likelihood.

For the following experiments, we estimate $\alpha = -1.298600417103578$ from both the training and test set in CIFAR10. We use binary search and set $\beta = 0.3763033370971679$ so that the FPR on CIFAR10 is 0.01%.

TABLE XIV
LIKELIHOOD-BASED DETECTOR AGAINST JB-TOP3

Query Set	**# Accounts**
CIFAR10	122
TinyImageNet	573
ImageNet1k	1
CIFAR100	514
SVHN	1
CINIC10	304
Indoor67	673
CUBS200	1
Caltech256	1

3) Effectiveness: One can tell from table XIV and table XV that likelihood-based detector outperforms similarity-based detector in terms of the number of adversarial accounts by 2×, but only when victim receive data from CIFAR10, where the likelihood-based detector was trained on. Otherwise, our likelihood-based detector either fails to detect any adversarial queries (*e.g.* for Caltech256, ImageNet1k, and SVHN), or the FPR is unacceptably high (*e.g.* for Indoor67, TinyImageNet, and CIFAR100). Figure 14 presents the changes in average likelihood as sending natural images in datasets, which further explain table XIV and table XV. These imply that although

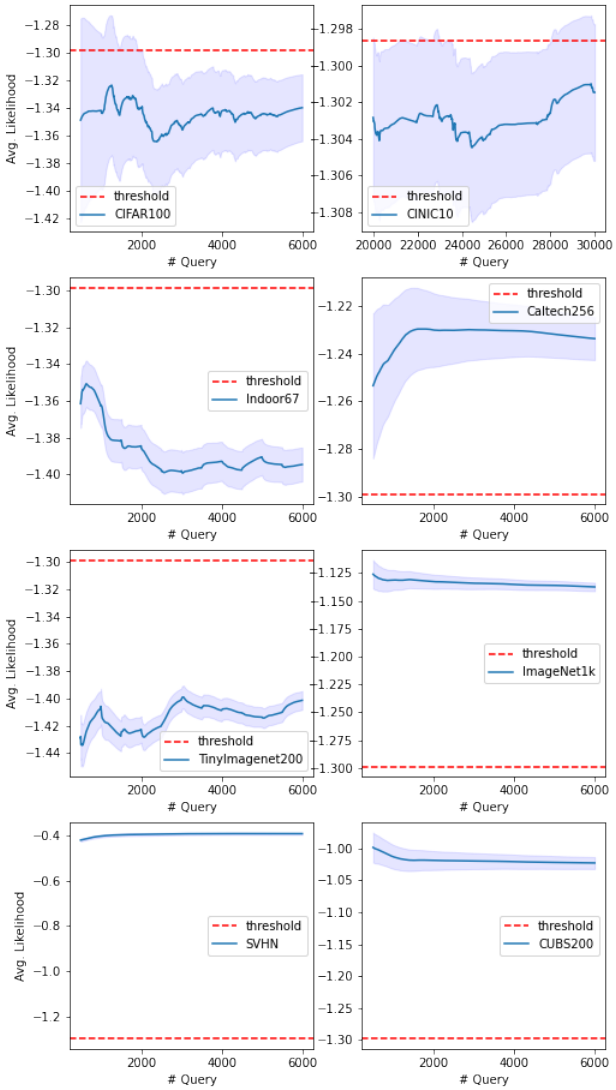

Fig. 14. Change in average log-likelihood over the number of queries. The credibility intervals of out-of-distribution (OOD) data often do not include the threshold.

likelihood-based detectors perform well in some dataset-specific scenarios, and there are at least two challenges to bring it into broader applications: 1) train the likelihood estimator (in our case, VQ-VAE) so that the average likelihood of benign queries converge to the value that is further away from adversarial ones', and 2) queries coming from closed problem-domain should have similar likelihood even though they come from

TABLE XV
FPR OF LIKELIHOOD-BASED DETECTOR

Query Set	FPR
CIFAR10	0.01%
TinyImageNet	1.45%
ImageNet1k	0%
CIFAR100	1.18%
SVHN	0%
CINIC10	0.17%
Indoor67	1.46%
CUBS200	0%
Caltech256	0.003%

different dataset (*e.g.* CIFAR10 and CINIC10).

On the other hand, table XIV and figure 14 suggest that if the defender rejects queries that are not in the credibility interval, then our VAE-based detector can possibly serve as a out-of-distribution (OOD) data detector. This property is helpful in a threat model where the defender assumes all OOD data are malicious ([44], [45]). However, we have not done further evaluation on whether our VAE-based detector can serve as a robust OOD detector, and we think this is an interesting direction for future work.

IX. CONCLUSION

In this paper, we explore Jacobian-based augmentation model extraction attack (JBA):

- JBA has advantage over other attacks when it is difficult for the attacker to collect large amount of data.
- We disprove the claim and assumption that Jacobian-based augmentation generates queries that cross or lie close to victim model's decision boundary.
- Complexity of query set helps improve JBA to some extent. The ideal query set should distribute as similarly to victim's training set as possible. For instance, a subset of victim's training set.

As a defender, we:

- build a similarity-based detector for detecting Jacobian-based model extraction attacks.
- propose *query filtering*, an adaptive attack against stateful detectors, and we show that our similarity-based detector can still work under query filtering attack.
- take the first step to design a VAE-based detector and identify some challenges with such defense scheme.

REFERENCES

[1] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," 2020. I, II-A, II-A, II-B, II-B

[2] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "Prada: Protecting against dnn model stealing attacks," 2019. I, II-A, III-A, III-A1, III-A2, IV, 1, IV-A2, IV-B, 1, IV-B2, V-A

[3] H. Yu, K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin, "Cloudleak: Large-scale deep learning models stealing through adversarial examples," in *NDSS*, 2020. I, III-A, III-A1, III-A2, III-A2, IV, IV-B3, IV-B3, 4, V-A

[4] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," 2017. I, II-A, III-A, III-A2, IV, 1, IV-B, IV-B2

[5] S. Chen, N. Carlini, and D. Wagner, "Stateful detection of black-box adversarial attacks," 2019. I, 2, II-B, V-A, V-D, VI-A, 9, VI-A, VII-A, VII-A, VII-A, VIII-A1, VIII-A1

[6] H. Li, S. Shan, E. Wenger, J. Zhang, H. Zheng, and B. Y. Zhao, "Blacklight: Defending black-box adversarial attacks on deep neural networks," 2020. I, II-B, V-D

[7] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," 2016. II-A, II-A, III-A, III-A3, III-B

[8] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data," *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul 2018. II-A, III-C

[9] N. Carlini, M. Jagielski, and I. Mironov, "Cryptanalytic extraction of neural network models," 2020. II-A, II-B

[10] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, (New York, NY, USA), p. 641–647, Association for Computing Machinery, 2005. II-A

[11] S. Milli, L. Schmidt, A. D. Dragan, and M. Hardt, "Model reconstruction from model explanations," 2018. II-A, II-B

[12] L. Batina, S. Bhasin, D. Jap, and S. Picek, "Csi neural network: Using side-channels to recover your artificial neural network information," 2018. II-A, II-B

[13] J. J. Douceur, "The sybil attack," in *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, January 2002. II-B

[14] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," *ACM Trans. Knowl. Discov. Data*, vol. 8, Feb. 2014. II-B

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2019. III-A2, IV-A2

[16] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2017. III-A2, IV-A2

[17] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," 2016. III-A2

[18] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference (BMVC)* (E. R. H. Richard C. Wilson and W. A. P. Smith, eds.), pp. 119.1–119.11, BMVA Press, September 2016. III-A2

[19] H. Karimi, T. Derr, and J. Tang, "Characterizing the decision boundary of deep neural networks," 2020. III-A3

[20] J.-B. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," 2021. III-B, VII

[21] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. III-B

[22] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, "Finite sample convergence rates of zero-order stochastic optimization methods.," in *NIPS*, pp. 1448–1456, Citeseer, 2012. III-B

[23] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," 2018. III-C

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. III-C

[25] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. Shevade, and V. Ganapathy, "Activethief: Model extraction using active learning and unannotated public data," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 865–872, Apr. 2020. III-C

[26] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR '94* (B. W. Croft and C. J. van Rijsbergen, eds.), (London), pp. 3–12, Springer London, 1994. III-C

[27] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," 2018. III-C

[28] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2016. III-C

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. IV-A1

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. IV-A1

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009. IV-A1, IV-A3, IV, V, VI

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. IV-A1

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017. IV-A2

[34] J. Rauber, W. Brendel, and M. Bethge, "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*, 2017. IV-A2

[35] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel, "Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax," *Journal of Open Source Software*, vol. 5, no. 53, p. 2607, 2020. IV-A2

[36] T. Orekondy, B. Schiele, and M. Fritz, "Prediction poisoning: Towards defenses against dnn model stealing attacks," in *International Conference on Learning Representations*, 2020. IV-A2, V-C

[37] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," 2015. IV-A3, IV, V, VI

[38] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. IV-A3, IV, V, VI

[39] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," *NIPS*, 01 2011. IV-A3, IV, V, VI

[40] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "Cinic-10 is not imagenet or cifar-10," 2018. IV-A3, IV, V, VI

[41] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420, 2009. IV-A3, IV, V, VI

[42] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011. IV-A3, IV, V, VI

[43] G. Griffin, A. Holub, and P. Perona, "Caltech256 image dataset," 2006. IV-A3, IV, V, VI

[44] B. G. Atli, S. Szyller, M. Juuti, S. Marchal, and N. Asokan, "Extraction of complex dnn models: Real threat or boogeyman?," 2020. V-B, VIII-B3

[45] S. Kariyappa and M. K. Qureshi, "Defending against model stealing attacks with adaptive misinformation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. V-B, VIII-B3

[46] S. Kariyappa, A. Prakash, and M. K. Qureshi, "Protecting {dnn}s from theft using an ensemble of diverse models," in *International Conference on Learning Representations*, 2021. V-C

[47] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 2137–2146, PMLR, 10–15 Jul 2018. V-D

[48] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2018. V-D

[49] S. Moon, G. An, and H. O. Song, "Parsimonious black-box adversarial attacks via efficient combinatorial optimization," 2019. V-D

[50] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," 2020. V-D

[51] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM Transactions on Graphics*, vol. 34, pp. 98:1–98:10, 07 2015. VI-A

[52] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," 2018. VI-B, 11, VIII-B2, VIII-B2

[53] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," 2016. VI-B