# ModelGuard: Information-Theoretic Defense Against Model Extraction Attacks

Minxue Tang and Anna Dai, *Duke University;* Louis DiValentin, Aolin Ding, and Amin Hass, *Accenture;* Neil Zhenqiang Gong, Yiran Chen, and Hai "Helen" Li, *Duke University*

https://www.usenix.org/conference/usenixsecurity24/presentation/tang

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

# ModelGuard: Information-Theoretic Defense Against Model Extraction Attacks

Minxue Tang[1], Anna Dai[1], Louis DiValentin[2], Aolin Ding[2], Amin Hass[2],
Neil Zhenqiang Gong[1], Yiran Chen[1], Hai "Helen" Li[1]
[1]*Department of Electrical and Computer Engineering, Duke University*
[2]*Cyber Security Lab, Accenture*
[1]{minxue.tang, anna.dai, neil.gong, yiran.chen, hai.li}@duke.edu
[2]{louis.divalentin, a.ding, amin.hassanzadeh}@accenture.com

## Abstract

Malicious utilization of a query interface can compromise the confidentiality of ML-as-a-Service (MLaaS) systems via model extraction attacks. Previous studies have proposed to perturb the predictions of the MLaaS system as a defense against model extraction attacks. However, existing prediction perturbation methods suffer from a poor privacy-utility balance and cannot effectively defend against the latest adaptive model extraction attacks. In this paper, we propose a novel prediction perturbation defense named ModelGuard, which aims at defending against adaptive model extraction attacks while maintaining a high utility of the protected system. We develop a general optimization problem that considers different kinds of model extraction attacks, and ModelGuard provides an information-theoretic defense to efficiently solve the optimization problem and achieve resistance against adaptive attacks. Experiments show that ModelGuard attains significantly better defensive performance against adaptive attacks with less loss of utility compared to previous defenses.

## 1 Introduction

With the rapid development of machine learning, more and more organizations have begun deploying well-trained machine learning models as black-box capabilities that provide services for customers, i.e. Machine-Learning-as-a-Service (MLaaS) systems. While easy access to well-trained models brings convenience to users and creates wealth for the model owner, it also leads to potential threats to the ownership and security of the confidential models deployed in MLaaS systems. Recent studies show that MLaaS systems are vulnerable to model extraction attacks, where the attacker can replicate the parameters or functionality from the deployed confidential (target) model to an extracted substitute model by querying the system and learning from the query result [21, 35, 43]. Model extraction attacks allow attackers to retain access to the service through the extracted substitute model while avoiding any subscription or access costs imposed by the model provider. Furthermore, the extracted substitute model can expose the weakness of the target model and makes it vulnerable to downstream attacks, such as evasion attacks and membership inference attacks [23, 38, 40, 41].

There have been many works discussing the defense against model extraction attacks, and the proposed defense methods mainly fall into four types: (a) Model extraction detection tries to distinguish the attacker from other benign users according to the distribution of the query data [23, 25]. (b) Model information monitoring tries to evaluate how much information of the target model has been exposed to a specific user, and adjust the response strategy according to the information gained by the user [13, 25]. (c) Model watermarking forces the target model to learn a non-removable "watermark" (a specific input-output pair) so that the owner can claim ownership with it [1, 22]. (d) Prediction perturbation perturbs the output (prediction) of the target model so that the attacker cannot extract the model correctly [28, 36].

On one hand, none of model extraction detection, model information monitoring, and model watermarking are general to all kinds of model extraction attacks. Model extraction detection is effective only when the query data of an attacker distributes differently from the query data of a benign user, which is a strong assumption in practice [35]. Model information monitoring is not able to defend against collaborative attackers who divide the query data among multiple users to avoid querying with a single user. And model watermarking cannot prevent attackers from stealing the model for private use or for subsequent attacks. On the other hand, although prediction perturbation makes the minimum assumption and is universal to all kinds of attacks, previous prediction perturbation methods usually suffer from a bad privacy-utility balance because of heuristic perturbation mechanisms. Additionally, recent studies have shown that there exist strong adaptive attacks that can significantly weaken the defensive performance of previous prediction perturbation methods [8].

In this paper, we propose a novel prediction perturbation method, ModelGuard, which aims at attaining a better privacy-utility balance under the presence of adaptive model extraction attacks. ModelGuard investigates a general optimization formulation that considers both parameter-stealing and functionality-stealing attacks. The optimization formulation aims at attaining the optimal defensive performance while maximally preserving the utility of the target model with the utility constraints. Since the adaptive attack method is arbitrary and unknown to the defender, we explore two different ways to deal with the adaptive attack when solving the optimization problem, resulting in two variants of ModelGuard: ModelGuard-W and ModelGuard-S. Our contributions are:

1. We propose the first general formulation for the defense against adaptive model extraction attacks. We unify the parameter-stealing attack and the functionality-stealing attack in our formulation such that we can defend against both kinds of attacks in one objective.

2. We develop a constrained optimization problem that aims at defending against adaptive model extraction attacks while maintaining the utility of the target model. We propose ModelGuard to solve the optimization problem with two variants: ModelGuard-W and ModelGuard-S. Especially, ModelGuard-S attains an information-theoretic defense against strong adaptive attacks and outperforms the other defenses.

3. We evaluate ModelGuard against a wide range of attacks, including a strong adaptive attack we propose based on the Bayesian estimator. We empirically show that compared with previous defenses, ModelGuard attains a significantly better privacy-utility balance when defending against adaptive model extraction attacks[1].

This paper is organized as follows: We define the threat model in Section 2 and we present our defense ModelGuard in Section 3. The experimental results are given in Section 4. We leave the discussion on related works to Section 5 and we conclude this paper in Section 6.

## 2 Threat Model

**Attacker's goal** Query-based model extraction attacks aim at learning a substitute model with the predictions returned by a black-box target model, as shown in Figure 1. We denote $f_t(\cdot;\boldsymbol{w}_t)$ as the target model with parameters $\boldsymbol{w}_t$ trained on a confidential dataset $(X_t, Y_t)$. When performing the model extraction attack, the attacker uses the unlabeled query dataset $X_q = [\boldsymbol{x}_{q,1}, \boldsymbol{x}_{q,2}, \cdots, \boldsymbol{x}_{q,N}] \in \mathbb{R}^{N\times d}$ (with $N$ samples in $d$ dimensions) to query the target model and get its predictions. If there is no prediction perturbation defense, the clean prediction set $Y_q = [\boldsymbol{y}_{q,1}, \boldsymbol{y}_{q,2}, \cdots, \boldsymbol{y}_{q,N}] = f_t(X_q;\boldsymbol{w}_t)$ will be returned to the attacker from the target model, where $\boldsymbol{y}_{q,i} \in \mathbb{R}^C$ is the prediction result (i.e., the confidence score vector over $C$ classes) of

[1]Get our codes at https://github.com/Yoruko-Tang/ModelGuard.

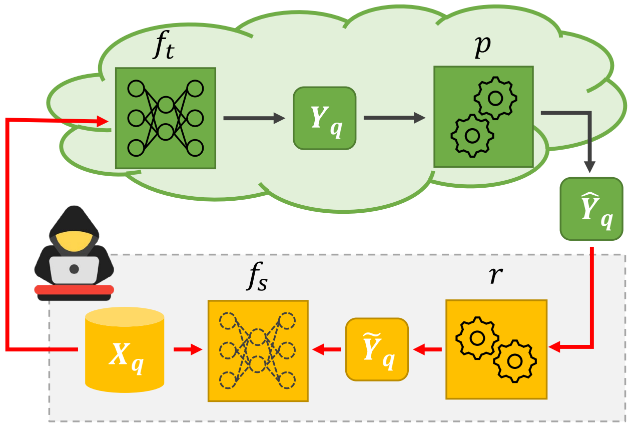

Figure 1: A general framework of a model extraction attack against a prediction perturbation defense. The clean predictions $Y_q$ of the query data $X_q$ outputted by the target model $f_t$ are perturbed to $\hat{Y}_q = p(Y_q)$ by a mechanism $p$, and the perturbed predictions are returned to the user. The adaptive attacker first tries to recover the clean predictions from the perturbed predictions with $\tilde{Y}_q = r(\hat{Y}_q)$. Then the attacker trains the substitute model $f_s$ with the recovered predictions.

the $i$-th query $\boldsymbol{x}_{q,i}$. Otherwise, the attacker will receive the perturbed prediction set $\hat{Y}_q = [\hat{\boldsymbol{y}}_{q,1}, \hat{\boldsymbol{y}}_{q,2}, \cdots, \hat{\boldsymbol{y}}_{q,N}] = p(Y_q)$ where $p$ is a perturbation mechanism. With the query data and the prediction returned from the target model, the model extraction attacker can train a substitute model $\boldsymbol{w}_s$ to attain two different goals: parameter-stealing and functionality-stealing.

For the parameter-stealing attack, the goal is to learn a substitute model with the same architecture and the same parameters as the target model, i.e., $\boldsymbol{w}_s^* = \boldsymbol{w}_t$. For the functionality-stealing attack, the goal is to learn the substitute model that can attain the maximum similarity with the target model in the clean predictions, namely,

$$\min_{\boldsymbol{w}_s} \quad L(X_q, Y_q; \boldsymbol{w}_s) = \mathcal{L}(f_s(X_q; \boldsymbol{w}_s), Y_q), \tag{1}$$

where $f_s(\cdot;\boldsymbol{w}_s)$ is the prediction of the substitute model $\boldsymbol{w}_s$, and $\mathcal{L}(Y_{\text{pred}}, Y_{\text{targ}})$ is a supervised loss that can measure how similar the predictions $Y_{\text{pred}}$ and the target labels $Y_{\text{targ}}$ are, e.g., cross-entropy (CE) loss or mean squared error (MSE) loss.

**Attacker's background knowledge** We assume that the attacker is not able to access the exact training dataset $(X_t, Y_t)$ of the target model. However, the attacker knows the domain of $X_t$ and can use any query data $X_q$ from a similar domain. Moreover, we consider a strong attacker who knows the model architecture of the target model so that the attacker is able to use the same model architecture in the substitute model when conducting the model extraction attack [8, 35].

In this paper, we assume that the attacker is adaptive, namely, the attacker knows about the prediction perturbation defense. In this case, the attacker first tries to recover the clean predictions $\tilde{Y}_q = r(\hat{Y}_q)$ from the perturbed predictions $\hat{Y}_q$ with a prediction recovery mechanism $r$. And then they use $\tilde{Y}_q$ to train the substitute model. We allow the adaptive attackers to know the details of the perturbation such that they can reproduce the same perturbation given a clean prediction [8].

*Remark* 1. Our adaptive attack model can cover the non-adaptive attack by simply setting $r$ to be an identity function. Additionally, an adaptive model extraction attack without an explicit prediction recovery can also be covered by our threat model. An arbitrary model extraction attack algorithm that results in a substitute model $\tilde{\boldsymbol{w}}_s^*$ satisfying $f_s(X_q;\tilde{\boldsymbol{w}}_s^*) = \tilde{Y}_q$ is equivalent to recovering the predictions to $\tilde{Y}_q = r(\hat{Y}_q)$ first and then training the substitute model with $(X_q, \tilde{Y}_q)$.

**Attacker's capability** We assume that the attacker can only interact with the target model by query, namely, there are no side channels providing extra information about the target model. The attacker can use any query data as long as it is a valid input of the target model, i.e., the attacker can use natural data or generate synthetic data for queries. And the attacker has unlimited query budgets, namely, the attacker can repeat querying with the same data arbitrarily many times. The attacker has sufficient computing power to implement any training and prediction recovery algorithms, and to implement the same perturbation mechanism as the defender.

## 3 ModelGuard Design

In this section, we introduce our defense framework, ModelGuard. We will first give a brief overview of ModelGuard in Section 3.1. We formulate our defense goal as a constrained optimization problem in Section 3.2, and then we introduce two variants of ModelGuard to solve the optimization problem from Section 3.3 to Section 3.5.

### 3.1 Overview

ModelGuard aims at defending both parameters and functionality of the target model against adaptive attacks while maintaining a high utility of the protected system. Our method is derived from a constrained optimization problem that tries to find the perturbed prediction set $\hat{Y}_q$ to maximize the loss $\mathcal{L}(\tilde{Y}_q, Y_q)$ under some utility constraints. However, since we are not aware of how the attacker recovers the prediction, it is intractable to directly solve the optimization problem. Instead, we explore two different ways to deal with the unknown recovery mechanism $r$ when solving the optimization problem, leading to two variant defense methods, ModelGuard-W and ModelGuard-S: ModelGuard-W assumes a weak adaptive attack and solves the optimization problem with the approximation $\tilde{Y}_q = r(\hat{Y}_q) \approx \hat{Y}_q$; ModelGuard-S considers the optimal recovery mechanism $r^*$ (i.e., the Bayes estimator) that leads to the lowest recovery loss, and we try to maximize an information-theoretic lower bound of the objective function against the attack with the optimal recovery mechanism.

### 3.2 Objective and Constraints

The goal of prediction perturbation is using the perturbed prediction set $\hat{Y}_q$ to prevent the attacker from replicating the target model parameters $\boldsymbol{w}_t$ or learning the functionality. To defend against parameter-stealing and functionality-stealing attacks discussed in Section 2, we can formulate two different optimization objectives respectively:

$$\max_{\hat{Y}_q} \quad \|\tilde{\boldsymbol{w}}_s^* - \boldsymbol{w}_t\|_2^2, \tag{2}$$

$$\max_{\hat{Y}_q} \quad L(X_q, Y_q; \tilde{\boldsymbol{w}}_s^*) = \mathcal{L}(\tilde{Y}_q, Y_q). \tag{3}$$

**A unified objective for defending against both attacks** Directly defending against parameter-stealing attacks is intractable since the defender does not know about the training details of the substitute model, such as the training hyperparameters and the training algorithms. The following lemma shows that maximizing $L(X_q, Y_q; \tilde{\boldsymbol{w}}_s^*)$ can also maximize the lower bound of $\|\tilde{\boldsymbol{w}}_s^* - \boldsymbol{w}_t\|_2^2$ under a smoothness assumption commonly used by optimization literature [5]. Therefore, we can use Equation (3) as a unified objective for defending against both kinds of attacks.

**Lemma 1.** *Assuming that $L(X_q, Y_q; \boldsymbol{w}) = \mathcal{L}(f_t(X_q; \boldsymbol{w}); Y_q)$ is $M$-smooth in $\boldsymbol{w}$. Given two models $\boldsymbol{w}_t$ and $\tilde{\boldsymbol{w}}_s^*$ with the same architecture such that $f_t(X_q; \boldsymbol{w}_t) = Y_q$ and $f_s(X_q; \tilde{\boldsymbol{w}}_s^*) = \tilde{Y}_q$, we have:*

$$\|\tilde{\boldsymbol{w}}_s^* - \boldsymbol{w}_t\|_2^2 \geq \frac{2}{M}\left[\mathcal{L}(\tilde{Y}_q, Y_q) - \mathcal{L}(Y_q, Y_q)\right]. \tag{4}$$

We prove Lemma 1 in Appendix A.1.

**Utility and validity constraints** We consider the scenarios where the users care about not only the top-1 label but also the confidence scores in the predictions, which can be used for downstream tasks such as out-of-distribution detection [18]. This leads to the following two utility constraints:

*(Distortion constraint)* As different downstream tasks may have different utility metrics, we adopt the $\ell_1$ norm of the perturbation as a generic metric of the utility loss [36]. And we constrain the distortion magnitude as follows:

$$\|\hat{\boldsymbol{y}}_{q,i} - \boldsymbol{y}_{q,i}\|_1 \leq \varepsilon, \quad i = 1, 2, \cdots, N. \tag{5}$$

*(Top-1 accuracy preserving constraint)* The top-1 label cannot be changed by the perturbation:

$$\arg\max_k \hat{y}_{q,i}^{(k)} = \arg\max_k y_{q,i}^{(k)} \quad i = 1, 2, \cdots, N, \tag{6}$$

where $\hat{y}_{q,i}^{(k)}$ is the $k$-th dimension of $\hat{\boldsymbol{y}}_{q,i}$.

In addition to the utility constraints, we also need to guarantee that the perturbed predictions are valid predictions. Thus we have the following validity constraint:

*(Simplex constraint)* The perturbed predictions must be valid confidence score vectors:

$$\sum_{k=1}^{C} \hat{y}_{q,i}^{(k)} = 1, \text{ and } \hat{\boldsymbol{y}}_{q,i} \succeq 0, \quad i = 1, 2, \cdots, N. \tag{7}$$

Namely, for each perturbed prediction, the sum of all dimensions is 1 and each dimension is non-negative. The $\ell_1$ distortion on $\hat{\boldsymbol{y}}_{q,i}$ cannot be larger than 2.0 under this constraint.

**Challenges in solving the optimization problem** It is still intractable to solve Equation (3) directly because we are not aware of what recovery mechanism $r$ is used to calculate $\tilde{Y}_q$. We explore two options to bypass this difficulty: (a) Assuming a weak adaptive attack and approximating $\tilde{Y}_q = r(\hat{Y}_q) \approx \hat{Y}_q$ when solving Equation (3); (b) Defending against the strongest adaptive attack where $r$ leads to the lowest $\mathcal{L}(\tilde{Y}_q, Y_q)$. In the following sections, we will introduce how to solve Equation (3) with these two options respectively.

## 3.3 MODELGUARD-W

In this section, we introduce the first variant MODELGUARD-W which adopts the first option, i.e., treats $\tilde{Y}_q \approx \hat{Y}_q$.

As we are focusing on classification tasks in this paper, we set $\mathcal{L}$ to be the CE loss $\mathcal{L}_{\text{CE}}$, which is one of the most popular loss functions in classification tasks. With this specific loss and the simplification $\tilde{Y}_q = \hat{Y}_q$, we can reformulate Equation (3) for each $i = 1, \cdots, N$ respectively as follows.

$$\begin{aligned} \min_{\hat{\boldsymbol{y}}_{q,i}} \quad & \sum_{k=1}^{C} y_{q,i}^{(k)} \log \hat{y}_{q,i}^{(k)} \\ \text{subject to} \quad & \|\hat{\boldsymbol{y}}_{q,i} - \boldsymbol{y}_{q,i}\|_1 \leq \varepsilon, \\ & \arg\max_k \hat{y}_{q,i}^{(k)} = \arg\max_k y_{q,i}^{(k)}, \\ & \sum_{k=1}^{C} \hat{y}_{q,i}^{(k)} = 1, \text{ and } \hat{\boldsymbol{y}}_{q,i} \succeq 0. \end{aligned} \tag{8}$$

Because of the logarithm in the objective function, it is difficult to solve this constrained non-convex optimization problem directly. A good convex approximation to this non-convex objective function is to swap $Y_q$ and $\hat{Y}_q$ in the loss function, which instead maximizes $\mathcal{L}_{\text{CE}}(Y_q, \hat{Y}_q)$ as follows:

$$\begin{aligned} \min_{\hat{\boldsymbol{y}}_{q,i}} \quad & \sum_{k=1}^{C} \hat{y}_{q,i}^{(k)} \log y_{q,i}^{(k)} \\ \text{subject to} \quad & \|\hat{\boldsymbol{y}}_{q,i} - \boldsymbol{y}_{q,i}\|_1 \leq \varepsilon, \\ & \arg\max_k \hat{y}_{q,i}^{(k)} = \arg\max_k y_{q,i}^{(k)}, \\ & \sum_{k=1}^{C} \hat{y}_{q,i}^{(k)} = 1, \text{ and } \hat{\boldsymbol{y}}_{q,i} \succeq 0. \end{aligned} \tag{9}$$

In this formulation, the objective function becomes linear w.r.t $\hat{\boldsymbol{y}}_{q,i}$, and thus this optimization problem becomes Linear Programming, which can be solved efficiently by existing algorithms [5]. We call this defense MODELGUARD-W.

*Remark* 2. As justified in Appendix A.2, the solution $\hat{\boldsymbol{y}}_{q,i}$ of Equation (9) assigns a small confidence score $\hat{y}_{q,i}^{(k)}$ to the class $k$ where $y_{q,i}^{(k)}$ is large, which can also lead to a small objective value in Equation (8). Thus, minimizing Equation (9) is an effective approximation of minimizing Equation (8).

MODELGUARD-W is effective against the attack that satisfies $\tilde{Y}_q \approx \hat{Y}_q$, but it may fail to defend against strong adaptive attacks where $\tilde{Y}_q$ significantly deviates from $\hat{Y}_q$. We will see how we are able to derive a defense against strong adaptive attacks with the optimal $r$ in the following sections.

## 3.4 Bayes Attack

Now we consider the second option, where we treat $r$ as the optimal prediction recovery that leads to the lowest loss $\mathcal{L}(\tilde{Y}_q, Y_q)$. We begin with the introduction to the optimal prediction recovery, i.e., the Bayes estimator, in this section. And we introduce the second variant, MODELGUARD-S, which defends the information-theoretic lower bound of the objective function in Section 3.5.

**Objective of the prediction recovery** The goal of the attacker is contrary to the goal of the defender given in Equation (3): the attacker tries to find a recovery function $r$ against the perturbation function $p$ that minimizes $\mathcal{L}(\tilde{Y}_q, Y_q)$. As the clean predictions are unknown to the attacker, $Y_q$ becomes a random variable in the perspective of the attacker, and the attacker tries to minimize the following posterior expected loss conditioned on the returned prediction set $\hat{Y}_q$:

$$\min_r \quad \mathbb{E}\left[\mathcal{L}(r(\hat{Y}_q), Y_q) | \hat{Y}_q\right], \tag{10}$$

where $\mathbb{E}$ stands for the expectation w.r.t $Y_q$.

For analysis simplicity, we adopt the most popular squared loss $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$ as the recovery loss[2], whose Bayes estimator (i.e., the minimizer $r^*$) is given by the posterior mean of $Y_q$ given $\hat{Y}_q$:

$$\tilde{Y}_q^* = r^*(\hat{Y}_q) = \mathbb{E}[Y_q | \hat{Y}_q]. \tag{11}$$

Namely, the optimal recovered prediction $\tilde{Y}_q^*$ is the average of all the clean predictions that can result in the same perturbed prediction $\hat{Y}_q$ returned to the attacker.

**Bayes Attack: a brute-force Bayes estimator** As the attacker does not know about the true distribution of $Y_q$, it is nontrivial to calculate the posterior mean $\mathbb{E}[Y_q | \hat{Y}_q]$ even though the

[2]Lemma 2 shows that our analysis can also adapt to CE loss.

attacker knows the defense mechanism. Previous adaptive attack methods use some approximate estimators as the solution to Equation (10), e.g., a trained neural network as an approximation of the Bayes estimator $r^*$ [8,28]. As an alternative, we propose a brute-force method which we call Bayes Attack, to find the Bayes estimator in Equation (11) under the assumption that the attacker has sufficient computing power. Since we allow the attacker to perform the same perturbation $p(Y)$ given any clean prediction set $Y = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N] \in \mathbb{R}^{N\times C}$, the attacker can build a lookup table that contains $(Y, p(Y))$ pairs of all possible valid $Y$. When given the perturbed prediction set $\hat{Y}_q$, the attacker finds all the $Y$ that satisfy $p(Y) = \hat{Y}_q$ in the table, then the mean of these $Y$ will be used as the recovered prediction set $\tilde{Y}_q$. We can summarize it as follows:

$$\begin{aligned} &\mathbb{T} = \{(Y, p(Y)) : \exists X \in \mathbb{R}^{N\times d}, \exists \boldsymbol{w}, Y = f_t(X; \boldsymbol{w})\}, \\ &\mathbb{M}(\hat{Y}_q) = \{Y : (Y, p(Y)) \in \mathbb{T}, \|p(Y) - \hat{Y}_q\| \le \delta\}, \\ &\tilde{Y}_q = \frac{1}{|\mathbb{M}(\hat{Y}_q)|} \sum_{Y\in\mathbb{M}(\hat{Y}_q)} Y, \end{aligned} \tag{12}$$

where $\mathbb{T}$ is the lookup table, and $\mathbb{M}(\hat{Y}_q)$ contains all matching clean prediction sets within a small error tolerance $\delta$.

*Remark* 3. Bayes Attack is proposed against the deterministic defense mechanism. For a randomized perturbation mechanism $p(Y)$, as we allow an unlimited query budget of the attacker, the attacker can repeat querying the target model with the same query data and obtain the mean of the perturbations. This makes it equivalent to attacking against a deterministic perturbation mechanism $p'(Y) = \mathbb{E}[p(Y)]$. Therefore, we exclude randomized perturbation from the discussion.

**Partial Bayes Attack: shrinking the sampling space** In practice, we are not able to implement the perfect Bayes Attack, because we do not have an infinite computing power to traverse all possible clean prediction sets $Y$, especially when we need to recover the prediction set $Y_q$ as a whole in a very high dimension ($Y_q \in \mathbb{R}^{N\times C}$). We can approximate the perfect Bayes Attack based on the fact that most defense mechanisms perturb each clean prediction $\boldsymbol{y}_{q,i}$ independently, i.e., $\hat{\boldsymbol{y}}_{q,i} = p(\boldsymbol{y}_{q,i})$ is only dependent on $\boldsymbol{y}_{q,i}$. Therefore, we can build a lookup table that contains $(\boldsymbol{y}, p(\boldsymbol{y}))$ pairs in a much lower dimension ($\boldsymbol{y} \in \mathbb{R}^C$) instead of $(Y, p(Y))$ pairs in a high dimension ($Y \in \mathbb{R}^{N\times C}$), namely,

$$\mathbb{T} = \{(\boldsymbol{y}, p(\boldsymbol{y})) : \exists \boldsymbol{x} \in \mathbb{R}^d, \exists \boldsymbol{w}, \boldsymbol{y} = f_t(\boldsymbol{x}; \boldsymbol{w})\}. \tag{13}$$

Even with the independent sampling, the sampling space of $\boldsymbol{y}$ is still huge when the number of classes is large (e.g., up to 256 in our experiment). Based on the observation that the prediction $\boldsymbol{y}$ in the neighborhood of each $\boldsymbol{y}_{q,i}$ is more likely to be perturbed to a result close to $\hat{\boldsymbol{y}}_{q,i}$, we further shrink the sampling space by only sampling clean predictions from the neighborhood of each true clean prediction $\boldsymbol{y}_{q,i}$ as elaborated in Appendix B.1. We obtain Partial Bayes Attack with the independent sampling and the neighborhood sampling.

## 3.5 ModelGuard-S

Now we introduce our second variant ModelGuard-S, which is an information-theoretic defense against strong adaptive attacks such as Bayes Attack. We target maximizing the expectation of the loss in Equation (3), i.e., $\mathbb{E}[\mathcal{L}(\tilde{Y}_q, Y_q)]$, against the prediction recovery with a statistical estimator. Given any recovery mechanism $r$, the expectation of the loss can be lower bounded by the following lemma derived from Information Theory:

**Lemma 2.** *Given a prediction perturbation mechanism $p$ such that $\hat{Y}_q = p(Y_q)$, an adaptive model extraction attack with an arbitrary recovery function $r$ cannot attain a smaller gap between recovered predictions $\tilde{Y} = r(\hat{Y}_q) = r(p(Y_q))$ and clean predictions $Y_q$ than the following lower bound:*

$$\mathbb{E}\left[\|\tilde{Y}_q - Y_q\|_2^2\right] \ge \frac{NC}{2\pi e} \exp\left(\frac{2}{NC} h(Y_q|\hat{Y}_q)\right), \tag{14}$$

*where $h(Y_q|\hat{Y}_q)$ is the conditional entropy. Subsequently,*

$$\mathbb{E}\left[\mathcal{L}(\tilde{Y}_q, Y_q) - \mathcal{L}(Y_q, Y_q)\right] \ge \frac{Cl}{2\pi e} \exp\left(\frac{2}{NC} h(Y_q|\hat{Y}_q)\right), \tag{15}$$

*where $l$ is a constant related to the loss function $\mathcal{L}$, e.g., $l =$ 0.5 for CE loss and $l = 1$ for MSE loss.*

We prove Lemma 2 in Appendix A.3.

Lemma 2 reveals that no adaptive attack can precisely recover the clean predictions when the uncertainty (i.e., the conditional entropy $h(Y_q|\hat{Y}_q)$) is still large given the perturbed predictions. Therefore, maximizing the conditional entropy can maximize the lower bound of the expected loss $\mathbb{E}[\mathcal{L}(\tilde{Y}_q, Y_q)]$ and defend against the adaptive attacks with any recovery mechanisms. Maximizing the conditional entropy is equivalent to minimizing the mutual information $I(Y_q; \hat{Y}_q)$ since $h(Y_q|\hat{Y}_q) = h(Y_q) - I(Y_q; \hat{Y}_q)$, so we can formulate our defense objective as follows:

$$\begin{aligned} \min_{\hat{Y}_q} \quad & I(Y_q; \hat{Y}_q) \\ \text{subject to} \quad & \|\hat{\boldsymbol{y}}_{q,i} - \boldsymbol{y}_{q,i}\|_1 \le \varepsilon, \\ & \arg\max_k \hat{y}_{q,i}^{(k)} = \arg\max_k y_{q,i}^{(k)}, \\ & \sum_{k=1}^{C} \hat{y}_{q,i}^{(k)} = 1, \text{ and } \hat{\boldsymbol{y}}_{q,i} \succeq 0, \\ & \text{for } i = 1, 2, \cdots, N. \end{aligned} \tag{16}$$

**Online prediction quantization** We notice that without the top-1 accuracy preserving constraint and the simplex constraint, Equation (16) is a rate-distortion problem in information theory. Since minimizing the mutual information in the rate-distortion problem is equivalent to finding the optimal

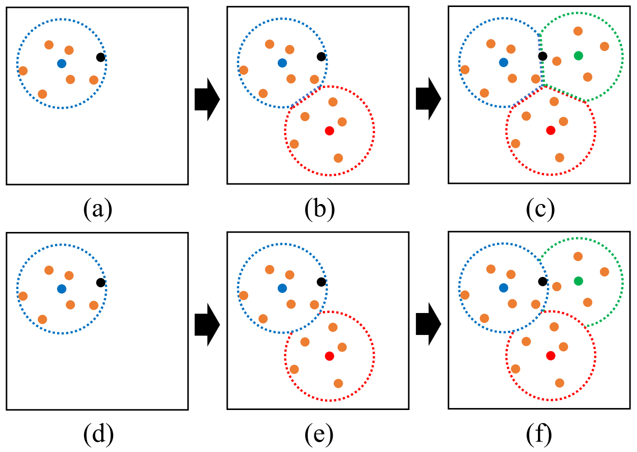
Figure 2: An illustration of non-ordered (upper) and ordered (lower) incremental prediction quantization. The order of the centroids is blue -> red -> green. The quantization result of the black point changes from the blue centroid in Figure (a) to the green centroid in Figure (c) in non-ordered incremental quantization, while it is always quantized to the blue centroid from Figure (d) to (f) in ordered incremental quantization.

compression method for the source data $Y_q$ under the distortion constraint [11], we can adopt a classical compression method, vector quantization, to solve this problem.[3] However, we are not able to use a standard vector quantization to compress the whole clean prediction set $Y_q$ because of the sequential characteristics of the query data and the requirement for a timely response. Namely, we have to perturb each prediction one by one as soon as it arrives instead of perturbing all predictions together after getting all the query data.

A naive solution to this problem is to quantize each clean prediction $\boldsymbol{y}_{q,i}$ independently with predefined static quantization centroids. Because static quantization does not utilize the distribution information of the source data, it usually either introduces a large distortion with a small number of centroids or attains a low compression rate (large mutual information) with a large number of centroids. Therefore, we turn to seek an online (dynamic) quantization method that can automatically increase the number of centroids to achieve a better balance between utility and defensive performance.

**Information leakage in online prediction quantization** A simple online quantization method is that we add a new centroid if no existing centroids are close enough to the incoming clean prediction. Otherwise, we quantize the incoming clean prediction to the closest centroid that satisfies the distortion constraint. However, we will see that this online quantization method may cause potential information leakage to the attacker, which may weaken the defensive performance.

[3]We do not consider randomized compression methods as discussed in Remark 3.

**Algorithm 1** ModelGuard-S
```
1: Initialize centroids $\mathbb{C}_i = \emptyset$ for each label $i = \{1,2,\cdots,C\}$.

2: while getting new input $\boldsymbol{x}_s$ from the user do
3:     Get clean prediction $\boldsymbol{y}_q = f_t(\boldsymbol{x}_s)$ with top-1 label $t$.
4:     if $\mathbb{C}_t == \emptyset$ or $\min_{\boldsymbol{c}\in\mathbb{C}_t} \|\boldsymbol{c}-\boldsymbol{y}_q\|_1 > \varepsilon$ then
5:         Append $\boldsymbol{y}_q$ to the end of $\mathbb{C}_t$.
6:     end if
7:     $\hat{\boldsymbol{y}}_q \leftarrow$ the first element in $\{\boldsymbol{c}\in\mathbb{C}_t : \|\boldsymbol{c}-\boldsymbol{y}_q\|_1 \le \varepsilon\}$.
8:     Return $\hat{\boldsymbol{y}}_q$ to the user.
9: end while
```

We consider quantizing the $t$-th clean prediction $\boldsymbol{y}_{q,t}$. We denote $Y_{q,t-1} = [\boldsymbol{y}_{q,1}, \boldsymbol{y}_{q,2}, \cdots, \boldsymbol{y}_{q,t-1}]$ as the historical clean perturbations, together with their perturbations $\hat{Y}_{q,t-1} = [\hat{\boldsymbol{y}}_{q,1}, \hat{\boldsymbol{y}}_{q,2}, \cdots, \hat{\boldsymbol{y}}_{q,t-1}]$ that have been returned to the user. We can decompose $h(Y_{q,t}|\hat{Y}_{q,t})$ as:

$$\begin{aligned} h(Y_{q,t}|\hat{Y}_{q,t}) =& h(Y_{q,t-1}, \boldsymbol{y}_{q,t}|\hat{Y}_{q,t-1}, \hat{\boldsymbol{y}}_{q,t}) \\ =& h(Y_{q,t-1}|\hat{Y}_{q,t-1}) + h(\boldsymbol{y}_{q,t}|Y_{q,t-1}, \hat{Y}_{q,t-1}) \\ & - I(Y_{q,t-1}; \hat{\boldsymbol{y}}_{q,t}|\hat{Y}_{q,t-1}) \\ & - I(\boldsymbol{y}_{q,t}; \hat{\boldsymbol{y}}_{q,t}|Y_{q,t-1}, \hat{Y}_{q,t-1}). \end{aligned} \quad (17)$$

Therefore, we need to minimize the last two terms to maximize the conditional entropy:

$$\min_{\hat{\boldsymbol{y}}_{q,t}} \quad I(\boldsymbol{y}_{q,t}; \hat{\boldsymbol{y}}_{q,t}|Y_{q,t-1}, \hat{Y}_{q,t-1}) + I(Y_{q,t-1}; \hat{\boldsymbol{y}}_{q,t}|\hat{Y}_{q,t-1}). \quad (18)$$

While the first term in Equation (18) can be minimized by an online quantization method that quantizes each clean prediction $\boldsymbol{y}_{q,t}$ one by one, the second term introduces an additional requirement that the new quantization result $\hat{\boldsymbol{y}}_{q,t}$ should not provide extra information about historical clean predictions $Y_{q,t-1}$ given previous quantization results $\hat{Y}_{q,t-1}$. For instance, given the same query data $\boldsymbol{x}$ with the same clean prediction $\boldsymbol{y}_{q,t_1} = \boldsymbol{y}_{q,t_2} = f_t(\boldsymbol{x})$, the quantization results at different times $t_1$ and $t_2$ should be consistent: $p(\boldsymbol{y}_{q,t_1}) = p(\boldsymbol{y}_{q,t_2}) = p(f_t(\boldsymbol{x}))$. Otherwise, the attacker can obtain new information about the clean prediction from the difference. While the online quantization increases the centroids over time to satisfy the distortion constraint, it may also introduce inconsistency in the quantization result. An example of such information leakage in online quantization is shown in the upper three figures of Figure 2, where we always quantize the new clean prediction to the nearest centroid. After the online quantization adds a new centroid (the green point) to satisfy the distortion constraint for new inputs coming in Figure (c), the quantization result of the black point changes from the blue point to the green point. This change allows the attacker to know that the green point is near the boundary of the blue and green areas in Figure (c) since the quantization result can be changed only when the point is near the boundary.

**MODELGUARD-S: an ordered incremental prediction quantization** Based on the analysis above, we propose the second variant, MODELGUARD-S, which is described in Algorithm 1 and illustrated in Figure 2 (d) - (f). The key idea of MODELGUARD-S is to automatically increase the number of centroids while maintaining the order of the centroids. For each class, MODELGUARD-S maintains one set of centroids for quantizing the predictions with the top-1 label of this class, such that MODELGUARD-S does not change the top-1 label after quantization (Line 1 in Algorithm 1). Given a new clean prediction for perturbing, if none of the existing centroids in the set are close enough to satisfy the distortion constraint, we append this clean prediction as a new centroid to the end of the set (Lines 4-5 in Algorithm 1). Each clean prediction is always quantized to the first centroid that satisfies the distortion constraint (Line 7 in Algorithm 1), such that we avoid the information leakage in different quantization results for the same data queried at different times.

An intuitive explanation of why prediction quantization can defend against adaptive attacks is that quantization maps all clean predictions in a cluster to the same perturbed prediction, i.e., the quantization centroid. Therefore, the attacker cannot distinguish between different clean predictions in the same cluster and the Bayes estimator becomes the mean of the cluster, which makes it unable to recover the clean prediction precisely given a perturbed prediction.

## 4 Experiments

In this section, we evaluate MODELGUARD and compare it with previous defense methods with extended experiments. We first introduce the common experiment settings in Section 4.1, then we show the defensive performance of different defense methods against a wide range of attack methods in Section 4.2. We will also give ablation studies to further explore and understand our MODELGUARD in Section 4.3

### 4.1 Experiment Setup

**Training datasets and model architectures of target models** In Table 1, we summarize the training datasets and the architectures of the target models. We conduct experiments with four target models trained on four different image classification datasets respectively: Caltech256 [15], CUB200 [45], CIFAR100, and CIFAR10 [26]. Since one of the baselines, Adaptive Misinformation [24], requires using Outlier Exposure (OE) [19] that trains the model with an additional OE dataset, we train all the target models used in our experiments with OE, adopting Indoor67 [39] and SVHN [33] as the OE dataset. We adopt two different models as the target models: ResNet50 [17] for Caltech256 and CUB200, and VGG16-BN [42] for CIFAR100 and CIFAR10. We also report the top-1 accuracy of each target model in Table 1.

**Query datasets and model architectures of substitute models** The query datasets and architectures of the substitute models are also summarized in Table 1. When conducting model extraction attacks, We use ImageNet1k [12] as the query dataset for target models trained on Caltech256 and CUB200, while we use TinyImageNet200 [32] as the query dataset for target models trained on CIFAR100 and CIFAR10. Following Orekondy et al. [36], we use the same model architecture of the target model for each substitute model.

**Evaluation metric** Following previous literature [24, 36], we mainly use the extraction accuracy, i.e., the top-1 accuracy of the substitute model tested on the test set of the target model, as the metric of defensive performance. We also use fidelity, i.e., the top-1 label agreement between the target model and the substitute model [28], as an auxiliary metric in a part of our experiments. The lower extraction accuracy and fidelity, the better defensive performance.

**Attack algorithms** Each model extraction attack algorithm can be decomposed into two parts: the query strategy and the attack strategy, and each query strategy can be combined with different attack strategies.

We consider two query strategies: KnockoffNet [35] and JBDA-TR [23]. The query dataset of KnockoffNet consists of only natural data, while JBDA-TR generates synthetic data from a small amount of natural data (called seed set) by using Jacobian-based data augmentation. We sample $50,000$ images as the query dataset for KnockoffNet and $1,000$ images as the seed set for JBDA-TR. We fix the random seed to sample the same images for all attack strategies and defense methods.

We consider the following attack strategies, including both weak adaptive attacks (Top-1 Attack, S4L Attack, and Smoothing Attack) and strong adaptive attacks (D-DAE Attack, D-DAE+ Attack, and Partial Bayes Attack):

1. Naive Attack: No prediction recovery is performed, and standard CE loss is used for training the substitute model.
2. Top-1 Attack: Only the hard top-1 label in the perturbed prediction is used for training the substitute model.
3. S4L Attack [21]: A self-supervised rotation loss is added to the CE loss for training the substitute model. S4L attack can be viewed as an adaptive attack without an explicit prediction recovery.
4. Smoothing Attack [29]: The query image is augmented by random affine augmentations. The predictions of several augmented images are averaged as the recovered prediction of the original image.
5. D-DAE [8]: A generative model is trained to recover the clean prediction. The training data (i.e.,(perturbed prediction, clean prediction) pairs) of this generative model is generated by small shadow models.

Table 1: Overview of experiment settings. The dataset and model architecture selections are partly following Orekondy et al. [36].

| | | | | | |
|---|---|---|---|---|---|
| Defender | Training Dataset $X_t$ | Caltech256 | CUB200 | CIFAR100 | CIFAR10 |
| | OE Dataset $X_{\mathrm{OE}}$ | Indoor67 | Indoor67 | SVHN | SVHN |
| | Target Model $\boldsymbol{w}_t$ | ResNet50 | ResNet50 | VGG16-BN | VGG16-BN |
| | Top-1 Accuracy of $\boldsymbol{w}_t$ | 85.67% | 82.19% | 75.11% | 93.70% |
| Attacker | Query Dataset $X_q$ | ImageNet1k | ImageNet1k | TinyImageNet200 | TinyImageNet200 |
| | Substitute Model $\boldsymbol{w}_s$ | ResNet50 | ResNet50 | VGG16-BN | VGG16-BN |

6. D-DAE+: D-DAE+ amends the original D-DAE method by using the lookup table generated for Partial Bayes Attack as the training data of the generative model.

7. Partial Bayes (pBayes) Attack: The independent sampling and the neighborhood sampling are used to establish the Bayes estimator as we proposed in Section 3.4.

**Defense methods** We evaluate the following defense methods in our experiments:

1. No defense (None): No perturbation is added to the prediction, and the clean prediction is always returned to the user.

2. Reverse Sigmoid Defense (RevSig) [28]: A reverse sigmoid function is used to perturb the clean prediction to attain large CE loss $\mathcal{L}_{\mathrm{CE}}(Y_q, \hat{Y}_q)$.

3. Maximizing Angular Deviation (MAD) [36]: The perturbation is obtained by maximizing the angular deviation between the gradient of the CE loss calculated with clean predictions $Y_q$ and the gradient of the CE loss calculated with perturbed predictions $\hat{Y}_q$.

4. Adaptive Misinformation (AM) [24]: An out-of-distribution (OOD) detection mechanism is combined with the prediction perturbation, and only the prediction of an OOD query data is perturbed to a wrong prediction generated from another misinformation function $\hat{f}_s$.

5. Top-1 Defense (Top-1): Only the hard top-1 label is returned to the user. Top-1 Defense can be viewed as a static quantization method that quantizes all the predictions with the same top-1 label to one centroid. Thus, Top-1 defense may introduce large distortions in the perturbed predictions with too few centroids.

6. Rounding Defense (Rounding): Only one decimal place is kept in each dimension of the prediction vector. Rounding defense can be viewed as another static quantization method that quantizes each dimension of the prediction independently. For example, $\boldsymbol{y} = [0.12, 0.31, 0.26, 0.31]$ is quantized as $\hat{\boldsymbol{y}} = [0.1, 0.3, 0.3, 0.3]$.

7. ModelGuard-W: The perturbations are calculated by solving Equation (9) as introduced in Section 3.3.

8. ModelGuard-S: The predictions are quantized as introduced in Section 3.5.

We provide more details of the above-mentioned attack and defense methods, as well as the model training hyperparameters in Appendix B.

## 4.2 Experimental Results

In this section, we conduct two groups of experiments. In the first group, we evaluate the performance of each defense method under fixed utility constraints to show that our method outperforms other baselines under the same utility constraint. In the second group, we show the privacy-utility trade-off of different defense methods with different distortion budgets, and we will see that our method attains a better balance between the defensive performance and the model utility compared with other methods.

**Defensive performance with fixed utility constraints** We first evaluate all the defenses under fixed utility constraints that keep $\ell_1$ distortion not more than 1.0 and top-1 accuracy not changed. More specifically, we fix $\varepsilon = 1.0$ for MAD, ModelGuard-W, and ModelGuard-S, and we fix $\gamma = 0.2$ and $\beta$ (0.008, 0.011, 0.02, 0.21 for Caltech256, CUB200, CIFAR100, and CIFAR10 respectively) for RevSig to make the maximal distortion (i.e., the largest distortion imposed by each defense in all perturbed predictions $\hat{\boldsymbol{y}}_{q,i} \in \hat{Y}_q$) at around 1.0 [4]. For AM, we fix $\tau$ (0.25, 0.3, 0.4, 0.7 for Caltech256, CUB200, CIFAR100, and CIFAR10 respectively) such that the protected accuracy (i.e., the accuracy of the target model $\boldsymbol{w}_t$ on the test set when adding perturbations to the predictions) does not drop significantly.

The attack and defense results on target models trained on Caltech256, CUB200, CIFAR100 and CIFAR10 are reported in Table 2, Table 3, Table 4 and Table 5 respectively. Each row reports the extraction accuracy of the substitute model using the specific attack method against the specific defense method given in each column. We report the maximal extraction accuracy and fidelity among all the attacks against each defense since we hope to verify each defense against the strongest attack. We also report the maximal $\ell_1$ distortion and the protected accuracy as the measurements of utility loss.

[4] We are not able to control the distortion of AM, Top-1 and Rounding Defense, thus we report their vanilla $\ell_1$ distortions.

Table 2: Defensive performance of different defense methods against different attacks on the target model trained on Caltech256.

| Query Method | Attack Method | None | RevSig | MAD | AM | Top-1 | Rounding | ModelGuard-W | ModelGuard-S |
|---|---|---|---|---|---|---|---|---|---|
| KnockoffNet | Naive Attack | 83.00% | 76.92% | 66.20% | 76.22% | 72.47% | 79.53% | 56.02% | 71.72% |
| | Top-1 Attack | 72.47% | 72.47% | 72.47% | 67.20% | 72.47% | 72.47% | 72.47% | 72.47% |
| | S4L Attack | 81.22% | 74.09% | 61.83% | 76.17% | 72.67% | 78.33% | 52.25% | 70.59% |
| | Smoothing Attack | 81.30% | 75.98% | 68.36% | 74.02% | 75.81% | 78.75% | 53.53% | 74.05% |
| | D-DAE | 83.00% | 80.76% | 80.16% | 77.38% | 73.23% | 75.20% | 50.97% | 74.28% |
| | D-DAE+ | 83.00% | 81.67% | 81.62% | 77.52% | 69.95% | 79.59% | 73.16% | 74.97% |
| | pBayes Attack | 83.00% | 82.81% | 82.80% | 82.78% | 68.31% | 81.76% | 80.83% | 75.17% |
| JBDA-TR | Naive Attack | 63.16% | 53.11% | 8.48% | 37.80% | 29.86% | 44.92% | 4.88% | 41.62% |
| | Top-1 Attack | 29.86% | 29.86% | 29.86% | 21.97% | 29.86% | 29.86% | 29.86% | 29.86% |
| | D-DAE | 63.16% | 54.23% | 17.89% | 37.44% | 26.02% | 40.39% | 3.06% | 40.70% |
| | D-DAE+ | 63.16% | 63.39% | 35.06% | 42.75% | 32.02% | 52.23% | 23.48% | 43.33% |
| | pBayes Attack | 63.16% | 62.55% | 36.26% | 62.86% | 31.25% | 54.78% | 25.73% | 42.80% |
| Max Accuracy of $\boldsymbol{w}_s$ | | 83.00% | 82.81% | 82.80% | 82.78% | 75.81% | 81.76% | 80.83% | **75.17%** |
| Max Fidelity of $\boldsymbol{w}_s$ | | 87.94% | 87.66% | 87.78% | 87.89% | **78.39%** | 86.16% | 85.00% | 79.05% |
| Max $\ell_1$ Distortion | | 0.00 | 1.01 | 1.00 | 2.00 | 1.99 | 1.00 | 1.00 | 1.00 |
| Protected Accuracy of $\boldsymbol{w}_t$ | | 85.67% | 85.67% | 85.67% | 84.16% | 85.67% | 85.67% | 85.67% | 85.67% |

Table 3: Defensive performance of different defense methods against different attacks on the target model trained on CUB200.

| Query Method | Attack Method | None | RevSig | MAD | AM | Top-1 | Rounding | ModelGuard-W | ModelGuard-S |
|---|---|---|---|---|---|---|---|---|---|
| KnockoffNet | Naive Attack | 71.28% | 57.71% | 49.90% | 52.09% | 49.10% | 63.27% | 32.14% | 52.97% |
| | Top-1 Attack | 49.10% | 49.10% | 49.10% | 27.27% | 49.10% | 49.10% | 49.10% | 49.10% |
| | S4L Attack | 66.21% | 49.86% | 43.46% | 44.80% | 45.58% | 56.16% | 25.16% | 48.03% |
| | Smoothing Attack | 66.08% | 53.18% | 48.84% | 49.12% | 53.33% | 59.27% | 29.77% | 52.36% |
| | D-DAE | 71.28% | 61.41% | 60.94% | 39.09% | 48.88% | 46.10% | 23.11% | 45.51% |
| | D-DAE+ | 71.28% | 68.93% | 69.16% | 53.43% | 29.01% | 63.91% | 58.97% | 53.07% |
| | pBayes Attack | 71.28% | 71.35% | 71.16% | 71.61% | 33.76% | 66.33% | 67.69% | 55.56% |
| JBDA-TR | Naive Attack | 28.48% | 19.90% | 0.59% | 5.25% | 11.48% | 10.72% | 0.54% | 12.84% |
| | Top-1 Attack | 11.48% | 11.48% | 11.48% | 2.05% | 11.48% | 11.48% | 11.48% | 11.48% |
| | D-DAE | 28.48% | 21.21% | 4.59% | 4.07% | 5.92% | 10.06% | 0.81% | 7.32% |
| | D-DAE+ | 28.48% | 27.91% | 7.80% | 19.16% | 3.12% | 14.38% | 2.86% | 10.10% |
| | pBayes Attack | 28.48% | 26.49% | 5.94% | 26.15% | 4.94% | 15.55% | 4.35% | 12.91% |
| Max Accuracy of $\boldsymbol{w}_s$ | | 71.28% | 71.35% | 71.16% | 71.61% | **53.33%** | 66.33% | 67.69% | 55.56% |
| Max Fidelity of $\boldsymbol{w}_s$ | | 79.67% | 79.20% | 78.86% | 79.70% | **57.59%** | 72.52% | 73.85% | 59.27% |
| Max $\ell_1$ Distortion | | 0.00 | 1.04 | 1.00 | 1.88 | 1.99 | 1.00 | 1.00 | 1.00 |
| Protected Accuracy of $\boldsymbol{w}_t$ | | 82.19% | 82.19% | 82.19% | 81.22% | 82.19% | 82.19% | 82.19% | 82.19% |

**pBayes Attack outperforms other attacks** Since we hope to verify our defense against the strongest adaptive attack, we first compare the results of different attacks. Among all the attacks, we can see that KnockoffNet + pBayes Attack attains the highest extraction accuracy in most cases. Previous studies have shown that when attacking against prediction perturbation defenses, model extraction attacks with natural data attain better results than attacks with synthetic data [8, 36]. And the superiority of pBayes Attack aligns with our analysis in Section 3.4 that pBayes Attack utilizes the Bayes estimator which can minimize the squared posterior expected loss. Some exceptions where pBayes Attack is not the best attack method could be attributed to the imperfect lookup table. Since we do not have the infinite computing power to generate the complete lookup table, a finite number of (perturbed predictions, clean predictions) pairs in the lookup table may introduce bias into the derived Bayes estimator.

**ModelGuard outperforms existing defenses** When defending against the strongest attack, Top-1 Defense and ModelGuard-S with strong prediction quantization always attain lower maximal extraction accuracy than the other defenses. While pBayes Attack can penetrate the other defenses and achieve a comparable extraction accuracy as attacking against no defense, none of the attacks can weaken the defensive performance of Top-1 Defense or ModelGuard-S to attain a significantly higher extraction accuracy than Naive Attack. This result is consistent with our analysis in Section 3.5 that no attacker can exactly recover the clean predictions from the quantized predictions because the quantization is not invertible. While Top-1 Defense achieves a little lower extraction accuracy compared with ModelGuard-S in some cases, it introduces a much larger $\ell_1$ distortion that exceeds the distortion constraint. Thus, ModelGuard-S should be chosen among all defenses we discuss since it is the strongest defense against strong adaptive attacks under utility constraints.

Although Rounding Defense also contains prediction quantization in its perturbation mechanism, it fails to achieve strong defensive performance because of static quantization. Namely, Rounding Defense introduces too many centroids

Table 4: Defensive performance of different defense methods against different attacks on the target model trained on CIFAR100.

| Query Method | Attack Method | None | RevSig | MAD | AM | Top-1 | Rounding | ModelGuard-W | ModelGuard-S |
|---|---|---|---|---|---|---|---|---|---|
| KnockoffNet | Naive Attack | 65.96% | 62.80% | 59.37% | 63.45% | 55.54% | 63.59% | 53.09% | 57.52% |
| | Top-1 Attack | 55.54% | 55.54% | 55.54% | 52.84% | 55.54% | 55.54% | 55.54% | 55.54% |
| | S4L Attack | 62.82% | 58.88% | 54.29% | 60.50% | 55.37% | 60.62% | 48.16% | 54.51% |
| | Smoothing Attack | 65.96% | 63.50% | 60.81% | 64.05% | 61.19% | 64.71% | 52.47% | 59.74% |
| | D-DAE | 65.96% | 63.61% | 63.09% | 61.93% | 57.14% | 62.68% | 49.38% | 59.10% |
| | D-DAE+ | 65.96% | 64.26% | 64.22% | 61.79% | 56.97% | 63.42% | 57.09% | 58.93% |
| | pBayes Attack | 65.96% | 65.44% | 65.25% | 65.59% | 57.15% | 64.83% | 62.54% | 58.67% |
| JBDA-TR | Naive Attack | 40.52% | 33.39% | 11.61% | 29.66% | 22.57% | 35.55% | 8.88% | 25.90% |
| | Top-1 Attack | 22.57% | 22.57% | 22.57% | 15.72% | 22.57% | 22.57% | 22.57% | 22.57% |
| | D-DAE | 40.52% | 29.63% | 12.73% | 25.96% | 17.44% | 25.32% | 3.76% | 22.43% |
| | D-DAE+ | 40.52% | 37.81% | 21.10% | 30.01% | 22.36% | 34.15% | 14.61% | 25.73% |
| | pBayes Attack | 40.52% | 39.03% | 26.49% | 39.73% | 23.22% | 38.11% | 21.15% | 25.94% |
| Max Accuracy of $\boldsymbol{w}_s$ | | 65.96% | 65.44% | 65.25% | 65.59% | 61.19% | 64.83% | 62.54% | **59.74%** |
| Max Fidelity of $\boldsymbol{w}_s$ | | 72.05% | 71.51% | 71.35% | 71.79% | 65.67% | 70.70% | 67.35% | **64.18%** |
| Max $\ell_1$ Distortion | | 0.00 | 1.00 | 1.00 | 1.99 | 1.98 | 1.00 | 1.00 | 1.00 |
| Protected Accuracy of $\boldsymbol{w}_t$ | | 75.11% | 75.11% | 75.11% | 74.30% | 75.11% | 75.11% | 75.11% | 75.11% |

Table 5: Defensive performance of different defense methods against different attacks on the target model trained on CIFAR10.

| Query Method | Attack Method | None | RevSig | MAD | AM | Top-1 | Rounding | ModelGuard-W | ModelGuard-S |
|---|---|---|---|---|---|---|---|---|---|
| KnockoffNet | Naive Attack | 87.32% | 84.86% | 83.61% | 82.84% | 83.51% | 86.91% | 75.06% | 84.11% |
| | Top-1 Attack | 83.51% | 83.51% | 83.51% | 80.30% | 83.51% | 83.51% | 83.51% | 83.51% |
| | S4L Attack | 85.99% | 82.04% | 80.72% | 81.86% | 83.72% | 85.49% | 71.23% | 82.80% |
| | Smoothing Attack | 87.86% | 85.27% | 84.07% | 84.81% | 86.16% | 87.37% | 76.26% | 85.36% |
| | D-DAE | 87.32% | 85.62% | 84.82% | 77.30% | 84.81% | 86.96% | 63.79% | 84.97% |
| | D-DAE+ | 87.32% | 86.58% | 86.84% | 84.17% | 84.26% | 86.93% | 58.23% | 84.27% |
| | pBayes Attack | 87.32% | 86.58% | 87.20% | 87.13% | 84.04% | 86.70% | 85.63% | 84.63% |
| JBDA-TR | Naive Attack | 75.50% | 66.54% | 53.32% | 59.56% | 62.55% | 73.38% | 38.13% | 61.55% |
| | Top-1 Attack | 62.55% | 62.55% | 62.55% | 53.71% | 62.55% | 62.55% | 62.55% | 62.55% |
| | D-DAE | 75.50% | 55.89% | 40.63% | 55.59% | 59.61% | 67.83% | 16.00% | 61.14% |
| | D-DAE+ | 75.50% | 72.48% | 67.45% | 65.59% | 63.28% | 72.67% | 31.86% | 64.65% |
| | pBayes Attack | 75.50% | 70.90% | 67.25% | 74.99% | 63.27% | 74.46% | 63.46% | 66.57% |
| Max Accuracy of $\boldsymbol{w}_s$ | | 87.86% | 86.58% | 87.20% | 87.13% | 86.16% | 87.37% | 85.63% | **85.36%** |
| Max Fidelity of $\boldsymbol{w}_s$ | | 89.72% | 89.20% | 89.73% | 89.56% | 88.12% | 89.47% | 87.97% | **87.14%** |
| Max $\ell_1$ Distortion | | 0.00 | 1.02 | 1.00 | 2.00 | 1.80 | 0.43 | 1.00 | 1.00 |
| Protected Accuracy of $\boldsymbol{w}_t$ | | 93.70% | 93.70% | 93.70% | 92.42% | 93.70% | 93.70% | 93.70% | 93.70% |

and fails to reduce the mutual information $I(Y_q;\hat{Y}_q)$.

We notice that when defending the target model trained on CIFAR10, there is only a small difference between different defense methods against strong adaptive attacks. Because of the small number of classes, the attacker can easily extract the target model even with only the hard label (i.e., Top-1 Attack). As a result, no defense method can defend against the model extraction attack effectively under the top-1 accuracy preserving constraint. In our result, ModelGuard-S attains the best defensive performance with the lowest maximal extraction accuracy and fidelity among all defenses, which verifies the effectiveness of ModelGuard-S in this extreme case.

**ModelGuard-W vs. ModelGuard-S** When defending against Naive Attack, Smoothing Attack and S4L Attack, ModelGuard-W attains the lowest extraction accuracy while satisfying both top-1 accuracy preserving constraint and distortion constraint. However, adaptive attacks with strong prediction recovery mechanisms, e.g., D-DAE+ and pBayes Attack, can significantly weaken the defensive performance of ModelGuard-W. We can see that pBayes Attack can penetrate ModelGuard-W and achieve a comparable extraction accuracy as attacking against no defense. This is because D-DAE+ and pBayes Attack significantly deviate from the assumption of ModelGuard-W that the attack is nearly non-adaptive, i.e., $\tilde{Y}_q \approx \hat{Y}_q$. In contrast, ModelGuard-S considers the strongest adaptive attack that can lead to the worst defensive performance, thus no attacks including D-DAE+ and pBayes Attack can attain significantly higher extraction accuracy than Naive Attack against ModelGuard-S. Since strong attacks should be considered by the defender, ModelGuard-S should be chosen among the two variants.

**Impact of the distortion budget** To evaluate the privacy-utility balance of different defenses, we now change the distortion budget in a range of $[0, 2.0]$, which covers all possible $\ell_1$ distortion values under the validity constraint. We set $\varepsilon \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0\}$ for MAD, ModelGuard-W and ModelGuard-S, and we vary $\beta \in [0, 1.0]$ with fixed $\beta = 0.2$ for RevSig to allow different distortions.

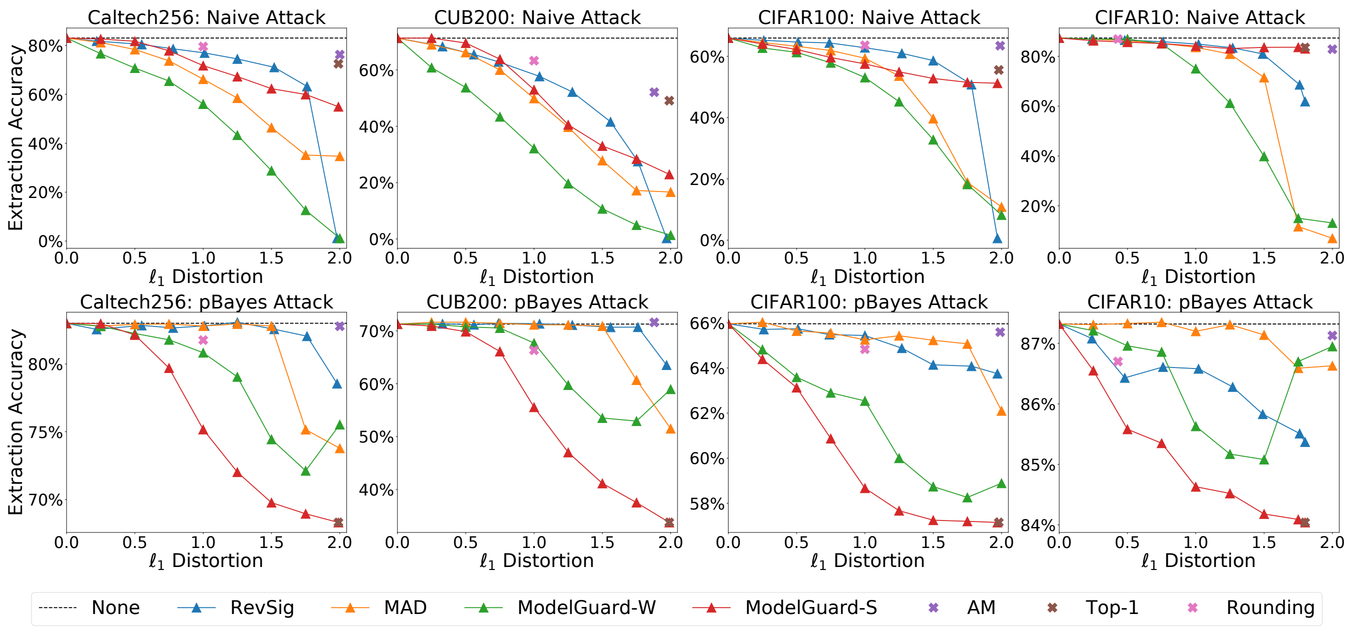

Figure 3: The extraction accuracy of Naive Attack (upper) and Partial Bayes Attack (lower) against defenses with different $\ell_1$ distortion budgets for the perturbation.

We show the extraction accuracy of Naive Attack and pBayes Attack against each defense in Figure 3.

As shown in the upper four figures, ModelGuard-W can attain the lowest extraction accuracy given different $\ell_1$ distortion budgets when defending against Naive Attack in most cases. MAD and RevSig can attain similar results with ModelGuard-W only when the distortion budget is very large (e.g., 2.0). The superiority of ModelGuard-W against Naive Attack is guaranteed by the optimality of Linear Programming when solving Equation (9), while previous methods are not able to guarantee the optimal defensive performance because of heuristic perturbation mechanisms.

Although ModelGuard-S does not attain a strong defense against Naive Attack because ModelGuard-S is maximizing the pessimistic lower bound of the objective function in Equation (3), it shows excellent resistance to pBayes Attack which utilizes a strong prediction recovery. ModelGuard-S consistently attains the lowest extraction accuracy among all the defenses under any distortion budget, and other defenses fail to defend against pBayes Attack until a large distortion budget (e.g., larger than 1.5). While Top-1 Defense lacks flexibility, ModelGuard-S can easily adapt to different distortion budgets and attain the best privacy-utility trade-off.

We notice that there are some fluctuations in the extraction accuracy of pBayes Attack when attacking against some defenses, which makes the extraction accuracy not monotonically decrease w.r.t the increase of the distortion budget. This is because the lookup table we use in pBayes Attack is randomly sampled, which inevitably introduces some variance in the recovery.

## 4.3 Ablation Studies

**How does conditional entropy influence defensive performance?** Since the conditional entropy $h(Y_a|\hat{Y}_a)$ generally decreases with the increasing number of centroids used for quantization, we plot how the extraction accuracy and the mean recovery distance $\mathbb{E}\|\tilde{\mathbf{y}}_q - \mathbf{y}_q\|_2$ of pBayes Attack change with respect to the number of centroids. We can see in Figure 4 that with more centroids used in quantization, i.e., smaller conditional entropy, the extraction accuracy increases while the recovery distance decreases. This result aligns with our theory that the adaptive attack can recover the prediction more precisely and attain higher extraction accuracy when the conditional entropy is smaller.

**How may a non-ordered incremental prediction quantization leak information?** As we discussed in Section 3.5, an online quantization that does not output consistent results given the same clean prediction may leak information to the attacker. We now show how this information leakage can happen in practice. We consider a simple attack strategy called *m* Queries Per Image (*m*-QPI), where the attacker repeats querying the target model with the same query dataset for *m* times and use the average of the query results as $\tilde{Y}_q$ to train the substitute model. For example, in 1-QPI, the attacker conducts only one query through the whole query dataset, which is the same as Naive Attack. In 2-QPI, the attacker conducts the second query after querying through the whole query dataset for the first time, and then the attacker averages the query results of the first query and the second query.

We consider the non-ordered incremental quantization that

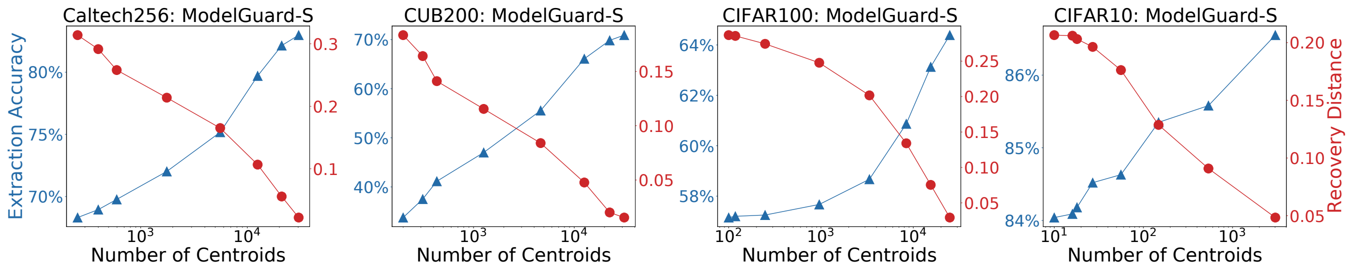

Figure 4: The extraction accuracy (blue line) and the mean recovery distance (red line) of Partial Bayes Attack against MODELGUARD-S with different numbers of centroids.

Table 6: The mean recovery distance of 1-QPI (1st column) and 2-QPI (2nd column) on the changed predictions. The 3rd column gives the number of predictions changed in the second query of 2-QPI.

| $X_t$ | 1-QPI | 2-QPI | Changed Predictions |
|---|---|---|---|
| Caltech256 | 0.300 | 0.193 | 2335 |
| CUB200 | 0.205 | 0.142 | 2443 |
| CIFAR100 | 0.372 | 0.239 | 2071 |
| CIFAR10 | 0.463 | 0.252 | 393 |

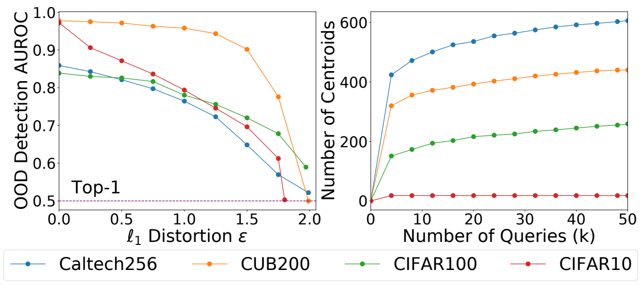

Figure 5: Left: AUROC of the OOD detector given predictions perturbed by MODELGUARD-S with different $\ell_1$ distortion budgets ε. Right: Number of centroids created by MODELGUARD-S along the query procedure.

always quantizes the clean prediction to the closest centroid. In Table 6, we report how many quantization results among total $50,000$ queries are changed in the second query of 2-QPI, together with the mean recovery distance $\mathbb{E}\|\tilde{\mathbf{y}}_q - \mathbf{y}_q\|_2$ of 1-QPI and 2-QPI on these changed predictions. We can see that the recovery distance can be significantly decreased by 2-QPI, which verifies that the non-consistent quantization results in a non-ordered incremental quantization may cause information leakage to the attacker.

We do not report the extraction accuracy here because the proportion of changed quantization results is too small (less than 5%) to cause a significant extraction accuracy change in our experiments. However, there still exists a potential risk of more significant information leakage in a larger system by using a non-ordered quantization. MODELGUARD-S with ordered quantization can completely eliminate such potential risk with consistent quantization results.

**How does MODELGUARD influence the utility in the downstream task?** While $\ell_1$ distortion in the prediction is a generic metric for measuring the utility loss of the perturbation, a specific utility metric may be considered for a specific downstream task when determining the distortion budget ε used by MODELGUARD. We consider a specific downstream task, out-of-distribution (OOD) detection, and we show how the $\ell_1$ distortion influences the AUROC of the OOD detector in Figure 5. Following [19], we use the Maximum Softmax Probability $\max_k y^{(k)}$ as the in-distribution (ID) score. We use $X_t$ as the ID set, and we use DTD [9] as the OOD set for Caltech256 and CUB200, while GTSRB [20] for CIFAR100 and CIFAR10. We can see that MODELGUARD-S can maintain a good detection performance (AUROC $\geq 0.8$) with $\varepsilon \leq 0.75$, while the detection performance is still acceptable (AUROC $\geq 0.7$ is acceptable according to [18]) until $\varepsilon = 1.5$ in most cases. Notice that Top-1 Defense always has AUROC $= 0.5$ because it only returns one-hot vectors for both ID and OOD queries.

**How does the independent sampling influence the performance of Partial Bayes Attack against MODELGUARD?** Since MODELGUARD-S utilizes dynamic quantization, the perturbation result depends on the historical queries that increase the number of centroids. However, we observe that MODELGUARD-S will behave like a static quantization method with almost invariant centroids after a short "warm-up" phase. Figure 5 shows the number of centroids used in MODELGUARD-S with $\varepsilon = 1.5$, and we can see that the centroids increase slowly after the first $4,000$ queries. This makes Partial Bayes Attack with independent sampling in Equation (13) attain similar performance with the perfect Bayes Attack that samples the whole prediction sequence in Equation (12) against MODELGUARD-S, while avoiding huge sampling cost for sampling in a high dimension.

**How efficient is MODELGUARD?** Since the real-time nature of an MLaaS system is usually important in many scenarios, the computational overhead introduced by the defense mechanism should be as little as possible. We compare the response time for returning the prediction of a single query to the user with different defenses in Figure 6. Among all de-

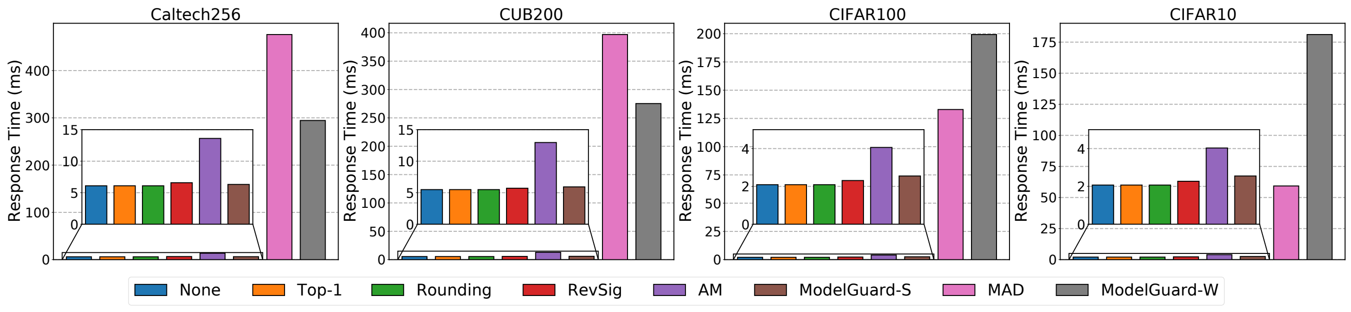

Figure 6: Response time of different defenses.

fenses, only MAD and MODELGUARD-W introduce a significant computational overhead when perturbing the prediction. MODELGUARD-S requires only marginal extra computation compared with None Defense since we only need to calculate the distance between the incoming clean prediction and the existing centroids when quantizing each prediction. Moreover, MODELGUARD-S only needs a small extra memory to store the quantization centroids. To summarize, MODELGUARD-S attains both high computation and memory efficiency.

## 5 Related Works

**Model extraction attacks** Machine Learning (ML) systems have been shown to be vulnerable to various kinds of attacks, such as evasion attacks [14], data poisoning attacks [3], backdoor attacks [16] and so on. In this paper, we focus on a specific attack called model extraction attack that aims at compromising the confidentiality of ML models. Model extraction attack aims at stealing the parameters or functionality of the target model deployed in the ML-as-a-Service (MLaaS) system such that the attacker can retain access to the service without paying [43]. Furthermore, the extracted substitute model can expose the vulnerability of the target model to the attacker such that the attacker can conduct powerful downstream attacks on the target model, such as white-box model evasion attacks [38], hyperparameter stealing attacks [46], and model inversion attacks [41]. Model extraction attacks can be conducted in different ways, e.g., via the power side channel [2] or via the query interactions [34], while we focus on query-based model extraction attacks in this paper.

Previous studies have shown that the attacker can use a small number of queries to attain high-fidelity model extraction, with either natural data [10,35] or synthetic data [23,44]. In addition, active learning [7] and semi-supervised learning [21] can also be applied in model extraction attacks to further improve query efficiency and attain higher extracted accuracy, which makes these attacks even more practical and threatening in realistic settings.

Recently, as more defenses against the model extraction attack have been proposed, adaptive model extraction attacks that can penetrate the defenses are getting more attention. Lee et al. [28] explore using a DNN model to recover the clean predictions given perturbed predictions from the target defended by a known defense mechanism. Chen et al. [8] complement this idea with a defense detector so that the attacker can detect an unknown defense and recover the clean predictions accordingly. In this work, we additionally propose a strong adaptive model extraction attack method based on the Bayesian estimator. We evaluate MODELGUARD and show its resistance against these adaptive attacks.

**Defenses against model extraction attacks** A wealth of works strive to defend the target against model extraction attacks and many different defense methods have been proposed. There are mainly four types of defenses:

(a) **Model extraction detection** detects the model extraction attacker based on the assumption that the query data from the attacker is distributed differently from the benign users [23,37]. Such detection can be bypassed by using query data that is similar to the data of benign users, e.g., natural data from a domain similar to the task domain of the target.

(b) **Model information monitoring** monitors the information of the model exposed to each user such that the system can reject to respond [25], add more noise in the response [47], or increase the query cost [13], for a user that may obtain too much information for model extraction. This kind of defense cannot defend against collaborative attacks, where each individual attacker queries the system with only a small amount of data, and all the attackers share the query results to collaboratively train a single substitute model.

(c) **Model watermarking** forces the target model to learn a specific input-output pair (i.e., watermark), such that the attacker cannot remove this "backdoor" when extracting the model, and the owner can claim the ownership with this non-removable watermark [1,4,6,22]. Model watermarking cannot prevent "stealth attackers" who only use the substitute model in private or for subsequent attacks.

(d) **Prediction perturbation** perturbs the prediction returned to the user to enlarge the loss [24,28] or change the gradient direction [30,36] calculated with the perturbed prediction, such that the attacker cannot extract the target correctly. Although prediction perturbation makes the least assumptions about the attacker, previous perturbation methods either achieve marginal defensive performance or impose a large utility loss because of heuristic perturbation mechanisms. In addition, there is little discussion about the prediction perturbation against adaptive attacks. In this work, we try to develop a novel prediction perturbation framework, ModelGuard, to attain a better privacy-utility trade-off. ModelGuard also provides the first information-theoretic prediction perturbation defense against adaptive model extraction attacks.

## 6 Conclusion, Limitations, and Future Work

In this paper, we propose ModelGuard, a novel prediction perturbation defense against adaptive model extraction attacks. ModelGuard investigates a general optimization problem that aims at defending against different kinds of model extraction attacks while maintaining a high utility of the protected system. We explore two different variants, ModelGuard-W and ModelGuard-S, to efficiently solve the optimization problem. Especially, ModelGuard-S utilizes an information-theoretic method that can defend against the adaptive attack with the optimal prediction recovery. Our experimental results show that ModelGuard-S attains a significantly better defensive performance than other baselines against adaptive attacks while maintaining a high utility.

Although ModelGuard-S is theoretically resistant to the perfect Bayes Attack, because of our limited computing power, we are not able to evaluate the defensive performance of ModelGuard-S against the perfect Bayes Attack with a complete lookup table. Our proposed Partial Bayes Attack samples a partial lookup table from the complete lookup table, which may cause a biased Bayes estimator and a suboptimal attack performance. It is a meaningful future work to derive a better estimation method for the perfect Bayes Attack.

Another important future direction is extending the defenses against model extraction attacks to domains beyond classification. While we only evaluate our defenses on image classification tasks following previous works [24, 36], our theory and methodology can adapt to different tasks by modifying the utility and validity constraints, as well as the loss function $\mathcal{L}$ in the objective function.

## Acknowledgements

We appreciate the constructive comments of the reviewers and the shepherd. This research is generously supported in part by Gift from Accenture, NSF CNS-2112562, CNS-2131859, CNS-2125977, CNS-1937786, CNS-1937787, and CNS-1822085, as well as ARO W911NF2110182.

## References

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

[2] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. Csi neural network: Using side-channels to recover your artificial neural network information. *arXiv preprint arXiv:1810.09076*, 2018.

[3] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[4] Franziska Boenisch. A survey on model watermarking neural networks. *arXiv preprint arXiv:2009.12153*, 2020.

[5] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *ACM Asia Conference on Computer and Communications Security*, 2021.

[7] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[8] Yanjiao Chen, Rui Guan, Xueluan Gong, Jianshuo Dong, and Meng Xue. D-dae: Defense-penetrating model extraction attacks. In *2023 IEEE Symposium on Security and Privacy (SP)*, 2022.

[9] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[10] Jacson Rodrigues Correia-Silva, Rodrigo F Berriel, Claudine Badue, Alberto F de Souza, and Thiago Oliveira-Santos. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018.

[11] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 2009.

[13] Adam Dziedzic, Muhammad Ahmad Kaleem, Yu Shen Lu, and Nicolas Papernot. Increasing the cost of model extraction with calibrated proof of work. *arXiv preprint arXiv:2201.09243*, 2022.

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[15] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[16] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[18] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[19] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[20] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, 2013.

[21] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *29th USENIX security symposium (USENIX Security 20)*, 2020.

[22] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. *arXiv preprint arXiv:2002.12200*, 2020.

[23] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019.

[24] Sanjay Kariyappa and Moinuddin K Qureshi. Defending against model stealing attacks with adaptive misinformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[25] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. Model extraction warning in mlaas paradigm. In *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018.

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

[27] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018.

[28] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. Defending against neural network model stealing attacks using deceptive perturbations. In *2019 IEEE Security and Privacy Workshops (SPW)*, 2019.

[29] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? In *2022 IEEE Symposium on Security and Privacy (SP)*, 2022.

[30] Mantas Mazeika, Bo Li, and David Forsyth. How to steer your adversary: Targeted and efficient model stealing defenses with gradient redirection. In *International Conference on Machine Learning*, 2022.

[31] Thomas Minka. Estimating a dirichlet distribution, 2000.

[32] Mohammed Ali mnmoustafa. Tiny imagenet, 2017.

[33] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[34] Seong Joon Oh, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[35] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[36] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. *arXiv preprint arXiv:1906.10908*, 2019.

[37] Soham Pal, Yash Gupta, Aditya Kanade, and Shirish Shevade. Stateful detection of model extraction attacks. *arXiv preprint arXiv:2107.05166*, 2021.

[38] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017.

[39] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE conference on computer vision and pattern recognition*, 2009.

[40] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

[41] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, 2017.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, 2016.

[44] Jean-Baptiste Truong, Pratyush Maini, Robert J Walls, and Nicolas Papernot. Data-free model extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[45] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[46] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *IEEE Symposium on Security and Privacy*, 2018.

[47] Haonan Yan, Xiaoguang Li, Hui Li, Jiamin Li, Wenhai Sun, and Fenghua Li. Monitoring-based differential privacy mechanism against query flooding-based model extraction attack. *IEEE Transactions on Dependable and Secure Computing*, 2021.

## A Proofs

### A.1 Proof of Lemma 1

**Lemma 1.** *Assuming that $L(X_q, Y_q; \boldsymbol{w}) = \mathcal{L}(f_t(X_q; \boldsymbol{w}); Y_q)$ is $M$-smooth in $\boldsymbol{w}$. Given two models $\boldsymbol{w}_t$ and $\tilde{\boldsymbol{w}}_s^*$ with the same architecture such that $f_t(X_q; \boldsymbol{w}_t) = Y_q$ and $f_s(X_q; \tilde{\boldsymbol{w}}_s^*) = \tilde{Y}_q$, we have:*

$$\|\tilde{\boldsymbol{w}}_s^* - \boldsymbol{w}_t\|_2^2 \geq \frac{2}{M}\left[\mathcal{L}(\tilde{Y}_q, Y_q) - \mathcal{L}(Y_q, Y_q)\right].$$

*Proof.* With the $M$-smoothness of $L(X_q, Y_q; \boldsymbol{w})$ and the fact that $\nabla L(X_q, Y_q; \boldsymbol{w}_t) = 0$, we have

$$\begin{aligned}\|\tilde{\boldsymbol{w}}_s^* - \boldsymbol{w}_t\|_2^2 &\geq \frac{2}{M}\left[L(X_q, Y_q; \tilde{\boldsymbol{w}}_s^*) - L(X_q, Y_q; \boldsymbol{w}_t)\right] \\ &= \frac{2}{M}\left[\mathcal{L}(\tilde{Y}_q, Y_q) - \mathcal{L}(Y_q, Y_q)\right],\end{aligned}$$

where the last equation comes from the definition that $f_t(X_q; \boldsymbol{w}_t) = Y_q$ and $f_t(X_q; \tilde{\boldsymbol{w}}_s^*) = f_s(X_q; \tilde{\boldsymbol{w}}_s^*) = \tilde{Y}_q$ since $f_t$ and $f_s$ has the same architecture. □

*Remark* 4. The smoothness assumption provides the relationship between the model parameters $\boldsymbol{w}$ and the loss $L$ such that we can unify the parameter-stealing attack and the functionality-stealing attack in one framework. However, MODELGUARD is still an effective defense against the functionality-stealing attack without the assumption, as justified by our experiments on neural networks that may not satisfy the smoothness assumption.

### A.2 Theoretical Justification of Approximating Equation (8) with Equation (9)

We will show that the solution $\hat{\boldsymbol{y}}$ of Equation (9) has a low confidence score $\hat{\boldsymbol{y}}^{(k)}$ at the class with large $\boldsymbol{y}^{(k)}$, which leads to a small objective value in Equation (8). To make the solution interpretable, we analyze the optimization problem given in Equation (9) with a distortion relaxation (i.e., $\varepsilon = 2.0$). Without loss of generality, we consider $y^{(1)} = \max_k y^{(k)}$.

$$\begin{aligned}\min_{\hat{\boldsymbol{y}}} \quad & \sum_{k=1}^{C} \hat{y}^{(k)} \log y^{(k)} \\ \text{subject to} \quad & \hat{y}^{(1)} \geq y^{(j)}, j = 2, 3, \cdots, C \\ & \sum_{k=1}^{C} \hat{y}^{(k)} = 1, \text{ and } \hat{\boldsymbol{y}} \succeq 0.\end{aligned}$$

The Lagrangian of this problem is given by:

$$\begin{aligned}&L(\hat{\boldsymbol{y}}, \mu_1, \cdots, \mu_C, \nu_2, \cdots, \nu_C, \lambda) = \\ &\sum_{k=1}^{C} (\log y^{(k)} - \mu_k + \lambda)\hat{y}^{(k)} - \sum_{j=2}^{C} \nu_j (\hat{y}^{(1)} - \hat{y}^{(j)}) - \lambda.\end{aligned}$$

The Karush-Kuhn-Tucker (KKT) condition for the optimal solution $(\hat{\boldsymbol{y}}, \mu_1, \cdots, \mu_C, \nu_2, \cdots, \nu_C, \lambda)$ of this problem is

$$\begin{cases} \hat{y}^{(1)} \geq \hat{y}^{(j)}, j = 2, \cdots, C; \sum_{k=1}^{C} \hat{y}^{(k)} = 1, \text{ and } \hat{\boldsymbol{y}} \succeq 0; \\ \mu_k \geq 0, k = 1, 2, \cdots, C; \nu_j \geq 0, j = 2, \cdots, C; \\ \partial L / \partial \hat{y}^{(1)} = \log y^{(1)} - \mu_1 + \lambda - \sum_{j=2}^{C} \nu_j = 0; \\ \partial L / \partial \hat{y}^{(j)} = \log y^{(j)} - \mu_j + \lambda + \nu_j = 0, j = 2, \cdots, C; \\ \mu_k \hat{y}^{(k)} = 0, k = 1, \cdots, C; \nu_j (\hat{y}^{(1)} - \hat{y}^{(j)}) = 0, j = 2, \cdots, C. \end{cases}$$

We first assume that $\nu_j = 0$, we get from the fourth condition:

$$\nu_j = 0 \Rightarrow \mu_j = \lambda + \log y^{(j)} \geq 0 \Rightarrow \lambda \geq -\log y^{(j)}.$$

Equivalently, with the last condition and the fact that $\hat{y}^{(1)} = \max_k \hat{y}^{(k)} > 0$, we have

$$\lambda < -\log y^{(j)} \Rightarrow \nu_j > 0 \Rightarrow \hat{y}^{(j)} = \hat{y}^{(1)} > 0.$$

On the other hand, if $\lambda > -\log y^{(j)}$, we know that

$$\lambda > -\log y^{(j)} \Rightarrow \mu_j - \nu_j = \lambda + \log y^{(j)} > 0 \Rightarrow \mu_j > \nu_j \geq 0,$$

which means that $\hat{y}^{(j)} = 0$ according to the last condition.

In the last case $\lambda = -\log y^{(j)}$, we have $\mu_j = \nu_j$. If $\mu_j = \nu_j > 0$, we have $\hat{y}^{(1)} = \hat{y}^{(j)} = 0$, which is impossible. Thus we have $\mu_j = \nu_j = 0$ and $0 \leq \hat{y}^{(j)} \leq \hat{y}^{(1)}$. In conclusion, for $j = 2, \cdots, C$, we have

$$\begin{cases} y^{(j)} < e^{-\lambda} \Rightarrow \hat{y}^{(j)} = \hat{y}^{(1)}; \\ y^{(j)} = e^{-\lambda} \Rightarrow 0 \leq \hat{y}^{(j)} \leq \hat{y}^{(1)}; \\ y^{(j)} > e^{-\lambda} \Rightarrow \hat{y}^{(j)} = 0. \end{cases}$$

This result shows that except for the top-1 class, all classes with $y^{(k)}$ larger than the threshold $e^{-\lambda}$ will be reduced to 0 in the solution $\hat{\boldsymbol{y}}$.

### A.3 Proof of Lemma 2

**Lemma 2.** *Given a prediction perturbation mechanism $p$ such that $\hat{Y}_q = p(Y_q)$, an adaptive model extraction attack with an arbitrary recovery function $r$ cannot attain a smaller gap between recovered predictions $\tilde{Y} = r(\hat{Y}_q) = r(p(Y_q))$ and clean predictions $Y_q$ than the following lower bound:*

$$\mathbb{E}\left[\|\tilde{Y}_q - Y_q\|_2^2\right] \geq \frac{NC}{2\pi e} \exp\left(\frac{2}{NC} h(Y_q | \hat{Y}_q)\right),$$

*where $h(Y_q | \hat{Y}_q)$ is the conditional entropy. Subsequently,*

$$\mathbb{E}\left[\mathcal{L}(\tilde{Y}_q, Y_q) - \mathcal{L}(Y_q, Y_q)\right] \geq \frac{Cl}{2\pi e} \exp\left(\frac{2}{NC} h(Y_q | \hat{Y}_q)\right),$$

*where $l$ is a constant related to the loss function $\mathcal{L}$, e.g., $l =$ 0.5 for CE loss and $l = 1$ for MSE loss.*

*Proof.* The following inequality holds for $Y_q$ with an arbitrary distribution conditioned on the event $\{\hat{Y}_q = \hat{Y}\}$ [11]:

$$h(Y_q|\hat{Y}_q = \hat{Y}) \le \frac{1}{2}\log\left((2\pi e)^{NC}\det(\mathrm{Cov}(Y_q|\hat{Y}_q = \hat{Y}))\right)$$
$$\Rightarrow \det\left(\mathrm{Cov}(Y_q|\hat{Y}_q = \hat{Y})\right) \ge \frac{1}{(2\pi e)^{NC}}\exp\left(2h(Y_q|\hat{Y}_q = \hat{Y})\right),$$

where the equality holds with Gaussian $Y_q|\{\hat{Y}_q = \hat{Y}\}$. Now we consider an arbitrary recovery function $r$ such that $\tilde{Y}_q = r(\hat{Y}_q)$. With the fact that the Bayesian estimator $\tilde{Y}_q^* = r^*(\hat{Y}_q) = \mathbb{E}[Y_q|\hat{Y}_q]$ minimizes $\mathbb{E}\left[\|\tilde{Y}_q - Y_q\|_2^2|\hat{Y}_q\right]$, we get:

$$\begin{aligned}
&\mathbb{E}\left[\|\tilde{Y}_q - Y_q\|_2^2|\hat{Y}_q = \hat{Y}\right] \ge \mathbb{E}\left[\|Y_q - \mathbb{E}[Y_q|\hat{Y}_q = \hat{Y}]\|_2^2|\hat{Y}_q = \hat{Y}\right]\\
=&\mathrm{tr}\left(\mathbb{E}\left[(Y_q - \mathbb{E}[Y_q|\hat{Y}_q = \hat{Y}])(Y_q - \mathbb{E}[Y_q|\hat{Y}_q = \hat{Y}])^T|\hat{Y}_q = \hat{Y}\right]\right)\\
=&\mathrm{tr}\left(\mathrm{Cov}(Y_q|\hat{Y}_q = \hat{Y})\right)\\
\ge& NC\left[\det\left(\mathrm{Cov}(Y_q|\hat{Y}_q = \hat{Y})\right)\right]^{1/NC} \qquad (19)\\
\ge&\frac{NC}{2\pi e}\exp\left(\frac{2}{NC}h(Y_q|\hat{Y}_q = \hat{Y})\right).
\end{aligned}$$

Inequality (19) uses the inequality of arithmetic and geometric means: $\forall \boldsymbol{A} \in \mathbb{S}_{++}^{n\times n}$,

$$\mathrm{tr}(\boldsymbol{A}) = \sum_i \lambda_i(\boldsymbol{A}) \ge n(\prod_i \lambda_i(\boldsymbol{A}))^{1/n} = n\left[\det(\boldsymbol{A})\right]^{1/n},$$

where $\lambda_i(\boldsymbol{A})$ is the $i$-th eigenvalue of $\boldsymbol{A}$. Taking expectation over $\hat{Y}_q$ at both sides yields the first part of the lemma with Jensen's Inequality.

Now we prove the second part of this lemma. For MSE loss, we notice that

$$\mathcal{L}_{\mathrm{MSE}}(\tilde{Y}_q, Y_q) = \frac{1}{N}\sum_{i=1}^{N}\|\tilde{\boldsymbol{y}}_{q,i} - \boldsymbol{y}_{q,i}\|_2^2 = \frac{1}{N}\|\tilde{Y}_q - Y_q\|_2^2$$

and $\mathcal{L}_{\mathrm{MSE}}(Y_q, Y_q) = 0$, thus we get $l = 1$ immediately.

For CE loss, We have

$$\mathcal{L}_{\mathrm{CE}}(\tilde{Y}_q, Y_q) - \mathcal{L}_{\mathrm{CE}}(Y_q, Y_q) = \frac{1}{N}\sum_{i=1}^{N} D_{\mathrm{KL}}(\boldsymbol{y}_{q,i}||\tilde{\boldsymbol{y}}_{q,i}).$$

$D_{\mathrm{KL}}(\boldsymbol{p}||\boldsymbol{q})$ is 1-strongly convex because

$$H_{\boldsymbol{p}} = \nabla_p^2 D_{\mathrm{KL}}(\boldsymbol{p}||\boldsymbol{q}) = \mathrm{diag}([1/p^{(1)}, 1/p^{(2)}, \cdots, 1/p^{(C)}]),$$

and $1/p^{(k)} \ge 1$ given $p^{(k)} \in [0,1]$. With $D_{\mathrm{KL}}(\boldsymbol{q}||\boldsymbol{q}) = 0$,

$$\begin{aligned}
D_{\mathrm{KL}}(\boldsymbol{p}||\boldsymbol{q}) &= D_{\mathrm{KL}}(\boldsymbol{p}||\boldsymbol{q}) - D_{\mathrm{KL}}(\boldsymbol{q}||\boldsymbol{q})\\
&\ge \left(\nabla_{\boldsymbol{p}} D_{\mathrm{KL}}(\boldsymbol{p}||\boldsymbol{q})\big|_{\boldsymbol{p}=\boldsymbol{q}}\right)^T(\boldsymbol{p}-\boldsymbol{q}) + \frac{1}{2}\|\boldsymbol{p}-\boldsymbol{q}\|_2^2\\
&= \sum_{k=1}^{C}(p^{(k)} - q^{(k)}) + \frac{1}{2}\|\boldsymbol{p}-\boldsymbol{q}\|_2^2 \qquad (20)\\
&= \frac{1}{2}\|\boldsymbol{p}-\boldsymbol{q}\|_2^2,
\end{aligned}$$

where Equation (20) uses the result that $\nabla_{\boldsymbol{p}} D_{\mathrm{KL}}(\boldsymbol{p}||\boldsymbol{q})\big|_{\boldsymbol{p}=\boldsymbol{q}} = [1,1,\cdots,1]$. Accordingly, we will get

$$\mathcal{L}_{\mathrm{CE}}(\tilde{Y}_q, Y_q) - \mathcal{L}_{\mathrm{CE}}(Y_q, Y_q) \ge \frac{1}{2N}\|\tilde{Y}_q - Y_q\|_2^2,$$

and $l = \frac{1}{2}$ for CE loss. □

## B Experiment Details

Our experiments are conducted on a server with one NVIDIA TITAN RTX GPU and one Intel Xeon Gold 6254 CPU.

### B.1 Attack Methods

**Query Strategies** We consider two query strategies:

**KnockoffNet [35]:** Following Orekondy et al. [36], we consider the KnockoffNet with a random strategy. The attacker randomly selects unlabeled natural data $X_q$ to query the target model and obtain the predictions $\hat{Y}_q$. $(X_q, \hat{Y}_q)$ then is used for training a substitute model with an attack strategy.

**JBDA-TR [23]:** JBDA-TR starts from a small query dataset (with 1,000 natural images in our setting) and repeatedly conducts two steps: (a) querying the target model and training a substitute model with the current query dataset; (b) augmenting the query dataset with a randomly targeted iterative Fast Gradient Sign Method (I-FGSM) [14, 27]:

$$X_{q,t} = X_{q,t-1} + \alpha\mathrm{sign}(\nabla f_s(X_{q,t-1}, Y_{q,t})), \quad t = 1,2,\cdots,T$$

where $\alpha$ is the step size and $T$ is the number of iterations in I-FGSM. $Y_{q,t}$ is randomly selected targeted labels in each iteration. The augmentation stops when the size of the query dataset reaches the targeted size ().

**Attack Strategies** Details of some attack strategies:

**S4L Attack [21]:** S4L Attack appends a self-supervised rotation loss to the CE loss, as given by:

$$\begin{aligned}
L_{\mathrm{S4L}}(X_q, \hat{Y}_q) =& \frac{1}{|X_q|}\sum_{\boldsymbol{x}_q \in X_q}\mathcal{L}_{\mathrm{CE}}(f_s(\boldsymbol{x}_q; \boldsymbol{w}_s), \hat{\boldsymbol{y}}_q)\\
&+ \frac{1}{4|X_q|}\sum_{\boldsymbol{x}_q \in X_q}\sum_{j=0}^{3}\mathcal{L}_{\mathrm{CE}}(f_r(R_j(\boldsymbol{x}_q); \theta_s), j),
\end{aligned}$$

where $R_j(\boldsymbol{x})$ rotates $\boldsymbol{x}$ by $j \times 90°$. $f_r(\cdot;\theta_s)$ shares the model with $f_s(\cdot;\boldsymbol{w}_s)$ except for the last layer.

**Smoothing Attack [29]:** Following Lukas et al. [29], we use random cropping and random horizontal flipping to generate $n = 3$ augmented images for each original image in the query dataset, and we average their query results as the recovered prediction of the original image.

**D-DAE [8] and D-DAE+:** We implement D-DAE without the defense detector since we assume that that attacker knows the details of the defense. D-DAE trains a three-layer neural network as the restorer to recover the clean predictions given the perturbed predictions. The training dataset of the restorer is $\{(p(f_i(\boldsymbol{x})), f_i(\boldsymbol{x})) : \boldsymbol{x} \in X_q, i = 1, \cdots, S\}$ where $f_i$ is a small shadow model trained on a public dataset (we use a randomly sampled subset of the query dataset with original labels as the public dataset). We generate a total of $1,000,000$ training samples with $S = 20$ shadow models in our experiments. D-DAE+ amends D-DAE by replacing its training dataset with the lookup table generated for Bayes Attack.

**Partial Bayes Attack:** Considering the limited computing resources we can use when evaluating our defenses, in addition to the independent sampling introduced in Equation (13), we further shrink the sampling space for the lookup table by only sampling possible clean predictions from the neighborhood of true clean predictions $\boldsymbol{y}_{q,i} \in Y_q$ as follows:

$$\mathbb{T} = \bigcup_{i=1}^{|Y_q|} \{(\boldsymbol{y}_j, p(\boldsymbol{y}_j)) : \boldsymbol{y}_j \sim \mathrm{Dir}(\boldsymbol{y}_{q,i}, s), j = 1, \cdots, K\}. \quad (21)$$

$\mathrm{Dir}(\boldsymbol{m}, s)$ is a Dirichlet distribution with mean $\boldsymbol{m}$, and the precision $s$ controls how concentrated the samples are [31]. We set $s = 8C$ and $K = 20$ in our experiments where $C$ is the number of classes, resulting in a lookup table with $1,000,000$ samples in total.

*Remark* 5. Notice that even the attacker uses $\boldsymbol{y}_{q,i} \in Y_q$ as the means, they do not know which one corresponds to a specific $\boldsymbol{x}_{q,i} \in X_q$. Therefore, the attacker is not able to use the means of the samples to train the substitute model directly. In addition, we set a relatively small precision $s$ in order to prevent the samples from becoming too close to the true clean prediction $\boldsymbol{y}_{q,i}$, which simulates the situation that the attacker does not know the clean predictions exactly.

## B.2 Defense Baselines

Details of some defense baselines are given as follows.

**Reverse Sigmoid (RevSig) [28]:** Lee et al. propose to use a Reverse Sigmoid function to perturb the prediction:

$$\hat{y}_{q,i}^{(k)} = \alpha_i \left[ y_{q,i}^{(k)} - \beta \left( s\left( \gamma s^{-1}(y_{q,i}^{(k)}) \right) - \frac{1}{2} \right) \right], \quad k = 1, \cdots, C.$$

$s(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, and $s^{-1}(\cdot)$ is its inverse function. $\alpha_i$ is a factor that normalizes $\hat{\boldsymbol{y}}_{q,i}$ such that $\sum_{k=1}^{C} \hat{y}_{q,i}^{(k)} = 1$. $\beta$ and $\gamma$ are hyperparameters that can control the perturbation magnitude.

**Maximizing Angular Deviation (MAD) [36]:** MAD maximizes the angle between the gradient calculated with $\hat{\boldsymbol{y}}_q$ and the gradient calculated with $\boldsymbol{y}_q$ as follows:

$$\max_{\hat{\boldsymbol{y}}_{q,i}} \left\| \frac{\boldsymbol{G}^T \hat{\boldsymbol{y}}_{q,i}}{\|\boldsymbol{G}^T \hat{\boldsymbol{y}}_{q,i}\|_2} - \frac{\boldsymbol{G}^T \boldsymbol{y}_{q,i}}{\|\boldsymbol{G}^T \boldsymbol{y}_{q,i}\|_2} \right\|_2^2$$

subject to (utility and validity constraints in Section 3.2)

$\boldsymbol{G}$ is the Jacobian of the substitute model: $\boldsymbol{G} = \nabla_{\boldsymbol{w}_s} \log f_s(\boldsymbol{x}_{q,i}; \boldsymbol{w}_s)$. As the defender is not aware of the substitute model $f_s(\cdot; \boldsymbol{w}_s)$, Orekondy et al. [36] propose to use a randomly initialized model $f_{s,\mathrm{sur}}(\cdot; \boldsymbol{w}_{s,\mathrm{sur}})$ as the surrogate model for the substitute model and calculate the Jacobian with this surrogate model during the defense.

**Adaptive Misinformation (AM) [24]:** AM is a selective prediction perturbation mechanism that relies on an out-of-distribution (OOD) detector. Their perturbation mechanism can be formulated as follows:

$$\begin{aligned} \hat{\boldsymbol{y}}_{q,i} &= (1-\alpha_i)\boldsymbol{y}_{q,i} + \alpha_i \hat{f}_t(\boldsymbol{x}_{q,i}; \boldsymbol{w}_p), \\ \text{where} \quad \alpha_i &= \frac{1}{1 + \exp\left( \nu(y_{q,i}^{(\max)} - \tau) \right)}. \end{aligned}$$

$\nu$ is a large constant (we set $\nu = 1000$ following Kariyappa et al. [24]) and $\hat{f}_t(\cdot; \boldsymbol{w}_p)$ is a misinformation model that is trained to minimize the reverse CE loss as follows:

$$\min_{\boldsymbol{w}_p} \quad \mathcal{L}_{\mathrm{CE}}(\mathbf{1} - \hat{f}_t(X_q; \boldsymbol{w}_p), Y_q),$$

where $\mathbf{1} = [1, 1, \cdots, 1]$ is a vector with all 1 elements.

## B.3 Training Hyperparameters

**Hyperparameters for training target models** We train the target models with Outlier Exposure (OE), which uses the following loss function:

$$\begin{aligned} \mathcal{L}_{\mathrm{OE}}(X_{\mathrm{ID}}, Y_{\mathrm{ID}}, X_{\mathrm{OOD}}; \boldsymbol{w}_t) = & \mathcal{L}_{\mathrm{CE}}(f_t(X_{\mathrm{ID}}; \boldsymbol{w}_t), Y_{\mathrm{ID}}) \\ & + \lambda \mathcal{L}_{\mathrm{CE}}(f_t(X_{\mathrm{OOD}}; \boldsymbol{w}_t), \mathcal{U}), \end{aligned}$$

where $(X_{\mathrm{ID}}, Y_{\mathrm{ID}})$ is equivalent to the in-distribution training dataset $(X_t, Y_t)$ (i.e., Caltech256, CUB200, CIFAR100 and CIFAR10), while $X_{\mathrm{OOD}}$ is the out-of-distribution dataset (i.e., Indoor67 and SVHN). $\mathcal{U}$ is a uniform label that has the same value at all dimensions. We set $\lambda = 1.0$ in our experiments following Kariyappa et al. [24].

We use models pretrained on ImageNet1k to initialize the target models in all our experiments. We train each model for 100 epochs. We use an SGD optimizer with an initial learning rate of 0.01, a batch size of 64, and a momentum of 0.5 for all the models, while we halve the learning rate every 30 epochs.

**Hyperparameters for training substitute models** We use the same pretrained model to initialize the substitute models as what we do for the target models. The number of training epochs is set to 30 for all the experiments as we find the substitute models can always converge within 30 epochs. We use an SGD optimizer with an initial learning rate of 0.01 and a batch size of 32 for Caltech256 and CUB200, and an initial learning rate of 0.1 and a batch size of 128 for CIFAR100 and CIFAR10. The momentum is also set to be 0.5. We halve the learning rate every 10 epochs.