Protecting Decision Boundary of Machine Learning Model With Differentially Private Perturbation

Huadi Zheng, *Student Member, IEEE*, Qingqing Ye, *Member, IEEE*, Haibo Hu, *Senior Member, IEEE*, Chengfang Fang, and Jie Shi

Abstract—Machine learning service API allows model owners to monetize proprietary models by offering prediction services to third-party users. However, existing literature shows that model parameters are vulnerable to extraction attacks which accumulate prediction queries and their responses to train a replica model. As countermeasures, researchers have proposed to reduce the rich API output, such as hiding the precise confidence. Nonetheless, even with response being only one bit, an adversary can still exploit fine-tuned queries with differential property to infer the decision boundary of the underlying model. In this article, we propose boundary differential privacy (BDP) against such attacks by obfuscating the prediction responses with noises. BDP guarantees an adversary cannot learn the decision boundary of any two classes by a predefined precision no matter how many queries are issued to the prediction API. We first design a perturbation algorithm called boundary randomized response for a binary model. Then we prove it satisfies ϵ-BDP, followed by a generalization of this algorithm to a multiclass model. Finally, we generalize a hard boundary to soft boundary and design an adaptive perturbation algorithm that can still work in the latter case. The effectiveness and high utility of our solution are verified by extensive experiments on both linear and non-linear models.

Index Terms—Model defense, boundary differential privacy, model extraction, adversarial machine learning

1 Introduction

The pervasive application of artificial intelligent has encouraged the boosting business of machine learning services, such as Microsoft Azure Face API, Google Cloud Speech-to-Text, and Amazon Comprehend. To train these high-quality machine learning models, service providers need to spend intense human labor and computation resources to acquire a large well-labeled datasets and tune training process. However, a prediction API call, which consists of a query and its response, can be vulnerable to adversarial attacks that disclose the internal states of these models. Particularly, a *model extraction* attack [1] is able to restore important model parameters using the rich information (e.g., model type, prediction confidence) provided by the prediction API. Once the model is extracted, an adversary can further apply model inversion attack [2] to learn the proprietary training data, compromising the privacy of data contributors. Another follow-up attack on the extracted model is *evasion attack* [3], [4], which avoids a certain prediction result by modifying its query. For example, a hacker modifies the executable binaries of a malware or the contents of a phishing email in order not to be detected by an antivirus or spam email filter.

Countermeasures against *model extraction* attacks have received increased attention but are still inadequate. One of them is to restrict rich information in the prediction API, for example, by rounding the prediction confidence value to a low granularity. However, even if the service provider completely eliminates this value in the prediction API, that is, to offer prediction label only, an adversary can still defeat this protection by issuing large number of fine-tuned queries and train a replica of the original model with great similarity [1], [3], [5]. The other countermeasure is to detect malicious extraction by monitoring feature coverage [6] or query distribution [7], and stop the service when a certain threshold is reached. However, since we cannot preclude user collusion, all queries and responses must be considered aggregately, which leads to significant false positive cases and eventually the early termination of service.

To address the disadvantages, in this paper we propose a new countermeasure that obfuscates the output label of a prediction response. There are three main concerns when designing this obfuscation mechanism. First, the accuracy of prediction API is highly correlated with the degree of obfuscation — if obfuscation needs to be applied to most queries, the utility of the machine learning service will degrade severely. Second, the obfuscation mechanism should be independent of the underlying machine learning models and can tackle a category of boundary-probing attacks. Third, the obfuscation mechanism should be customizable. That is, it should allow user-defined parameters that can trade utility for model privacy or vice versa.

- *Huadi Zheng, Qingqing Ye, and Haibo Hu are with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong, and also with the PolyU Shenzhen Research Institute, Shenzhen, Guangdong 518000, China. E-mail: huadi.zheng@connect.polyu.hk, {qqing.ye, haibo.hu}@polyu.edu.hk.*
- *Chengfang Fang and Jie Shi are with the Huawei International, Singapore 486066. E-mail: {fang.chengfang, shi.jie1}@huawei.com.*

Manuscript received 18 Nov. 2019; revised 1 Dec. 2020; accepted 6 Dec. 2020. Date of publication 8 Dec. 2020; date of current version 13 May 2022. (Corresponding author: Haibo Hu)
Digital Object Identifier no. 10.1109/TDSC.2020.3043382

Our key observation is that many model extraction attacks exploit fine-tuned queries near the decision boundary of a machine learning model achieve optimal extraction performance [8], [9]. We treat decision boundary probing as an abstract and necessary condition of high-quality extraction attacks. The responses of these queries disclose the details of model parameters and therefore should be obfuscated with priority. To this end, we propose a **b**oundary **d**ifferentially **p**rivate **l**ayer (BDPL) for machine learning services. BDPL provides a parameterized approach to obfuscate responses whose queries fall in a predefined boundary-sensitive zone. The notion of differential privacy guarantees the responses of all queries in the boundary-sensitive zone are indistinguishable from one another. As such, adversary cannot learn the decision boundary no matter how many queries are issued to the prediction API. On the other hand, the majority perturbation falls within boundary-sensitive zone and out-of-zone query is less affected from obfuscation. In this way, we can make the best use of the obfuscation and retain high utility of the machine learning service. To summarize, our contributions in this paper are as follows.

- We propose a new protection mechanism, namely, boundary differential privacy, against model extraction with fine-tuned queries while balancing service utility and model protection level.
- We develop an efficient method to identify queries in the boundary-sensitive zone, and design a perturbation algorithm called boundary randomized response for binary model to guarantee boundary differential privacy.
- We generalize binary defense to multiclass model and develop corresponding perturbation algorithm in a pairwise manner.
- We design an alternative defense layer with soft margin to extend the scope of protection and implement an adaptive perturbation algorithm.
- We conduct extensive empirical study on both binary and multiclass, linear and non-linear machine learning models to evaluate the effectiveness of our solution.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries for machine learning, model extraction and differential privacy. Section 3 elaborates on the threat model and problem definition with boundary-sensitive zone and boundary differential privacy. Section 4 presents the details of boundary differentially private layer. Section 5 introduces evaluation metrics and shows the experimental results of BDPL against model extractions. Section 7 reviews the related literature, and Section 8 concludes this paper and discusses future work.

2 Preliminaries

2.1 Supervised Machine Learning Model

A dataset $\mathcal{X}$ contains samples in a d-dimensional feature space. Each sample has a membership in a set of predefined classes called *labels*. Supervised machine learning trains a statistical model by such sample-label pairs to make predictions of labels on unknown samples. In this paper we focus on classification models which have K possible outputs. Formally, a classification model f produces a response y to a query sample $\boldsymbol{x}$ as follows.

$$y = f(\boldsymbol{x}) = \begin{cases} \text{''class 1'' label} \\ \text{''class 2'' label} \\ \quad\quad ... \\ \text{''class K'' label} \end{cases}.$$

Classification models have been widely adopted in many machine learning applications, such as activity categorization, face recognition and speaker identification. Depending on the nature of these applications, the model f can be either linear (e.g., logistic regression) or non-linear (e.g., neural network).

2.2 Model Extraction With Only Labels

In a model extraction attack, a malicious party attempts to replicate a model from the original one by continuously exploiting the prediction API. Technically any queries can constitute such an attack. However, the more queries the more likely this malicious attack will be exposed. As such, in the literature most model extraction attacks fabricate *fine-tuned queries* by differential techniques such as line search [1], [5] and Jacobian augmentation [3]. These queries are carefully selected to capture the information about decision boundary where prediction results vary drastically.

Formally, a model extraction attack selects a set of fine-tuned queries $\mathcal{X}_{diff}$ and obtains their responses $\mathcal{Y}_{diff}$ to train a replica model f'.

$$\begin{aligned} &\mathcal{X}_{diff} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}, \quad \boldsymbol{x} \in \mathbb{R}^d, \\ &\mathcal{Y}_{diff} = \{y_1, y_2, \ldots, y_n\}, \quad y \in \mathbb{R}^1, \\ &\exists \boldsymbol{x},\ \boldsymbol{x'} \in \mathcal{X}_{diff},\ dist(\boldsymbol{x}, \boldsymbol{x'}) = \delta\ \wedge\ y \neq y', \end{aligned}$$

where $dist(\cdot)$[1] measures the distance between two queries and δ is the unit distance adopted in the differential techniques when searching for boundary, i.e., where two corresponding responses $y \neq y'$.

2.3 Differential Privacy

Differential privacy [10], [33], [34] is proposed to bounds data sanitation with a measurable budget so that sensitive information can be released with a strong privacy guarantee. In centralized sanitation, all sensitive data are processed in one place and it is primarily defined in terms of adjacent datasets that differ on one data point with each other.

A perturbation algorithm $A(\cdot)$ probabilistically modifies the original data to other values in the same domain. It achieves ϵ-differential privacy, if and only if for any two adjacent datasets D, D' and any possible output τ of the perturbation algorithm, the following inequality always holds.

$$e^{-\epsilon} \leq \frac{Pr[A(D) = \tau]}{Pr[A(D') = \tau]} \leq e^{\epsilon}$$

Intuitively, privacy budget ϵ controls how close the sanitized data is to the original one. A larger privacy budget will induce a higher degree of similarity as well as utility.

1. In general, this notation can be any distance metrics (e.g., Manhattan distance, euclidean distance). The implications of distance metrics to detailed algorithms will be discussed in Section 4.1.1.

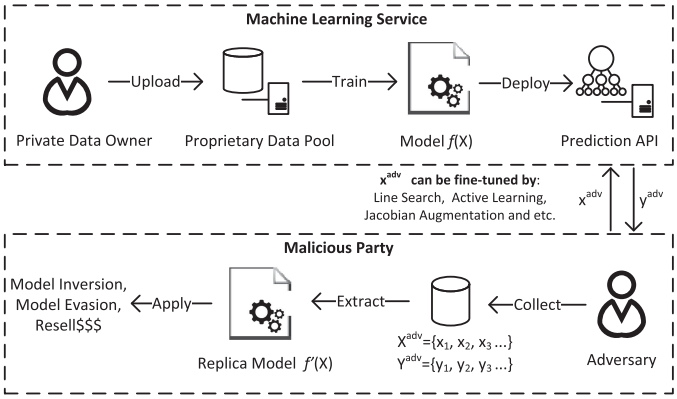


Fig. 1. Motivation and threat model.

3 Problem Definition

3.1 Motivation and Threat Model

A machine learning service provides a prediction result using a proprietary model as shown in Fig. 1. An adversary wants to produce a replica of this model by continuously querying it through the provided prediction API. We assume he can perform a typical two-stage extraction attack: 1) The adversary generates a set of fine-tuned real/synthetic queries normalized in [-1,1] and interacts with API under a large query budget. 2) He can store all responses, i.e., labels and reconstruct a replica model by training on the fine-tuned query-response pairs (e.g., minimize the empirical risk of an objective function). The success replication will results in intellectual property loss for the original provider and induce other attacks. The attack is semi-whitebox,[2] i.e., he can extract a replicated model using the same model type (e.g., convolutional neural network) and hyperparameters as the original one. Apart from public knowledge of model and defense settings, we assume the adversary has no apriori information of the decision boundary.

3.2 Boundary-Sensitive Zone

Our problem is to protect against model extraction attacks by obfuscating query responses. Before we formally define the security model, we first introduce the notion of *decision boundary* and *boundary-sensitive zone*. For most supervised models, a decision boundary is a critical borderline in the feature space where labels are different on both sides. Fig. 2 illustrates the hypothetical decision boundaries in four combination cases in a 2D feature space. In a multi-dimensional feature space, a line boundary becomes a hyperplane, and a curve boundary becomes a hypersurface.

Our key idea is to protect the query responses near the decision boundary against most model extraction attacks. To this end, we introduce the notion of boundary-sensitive zone.

Definition 1 (Boundary-Sensitive Zone). *Given feature space Z, a model f and a parameter Δ chosen by the model owner, all feature vectors adjacent to the decision boundary of f constitute a subspace Z_Δ of Z, where*

$$Z_\Delta = \{\boldsymbol{x} \in \mathbb{R}^d \mid dist(\boldsymbol{x}, f) < \Delta\},$$

where $dist(\cdot)$ measures the distance between a feature vector $\boldsymbol{x}$ and the decision boundary of f. All queries in this zone Z_Δ are considered particularly sensitive and have high risk of revealing the decision boundary of this model.

3.3 Boundary Differential Privacy

All queries in the boundary-sensitive zone need obfuscation, whose objective is to perturb the responses of any two sensitive queries so that they are indistinguishable for the adversary to determine the true decision boundary within this zone. To this end, we adopt the notion of differential privacy and formally define *boundary differential privacy* as follows.

Definition 2 (ϵ-Boundary Differential Privacy). *A perturbation algorithm $A(\cdot)$ achieves ϵ-boundary differential privacy, if and only if for any two queries $\boldsymbol{x}_1, \boldsymbol{x}_2$ in the boundary-sensitive zone Z_Δ, the following inequality always holds for the true responses y_1 and y_2 and the perturbed ones $A(y_1)$ and $A(y_2)$.*

$$e^{-\epsilon} \leq \frac{Pr[y_1 = y_2 | A(y_1), A(y_2)]}{Pr[y_1 \neq y_2 | A(y_1), A(y_2)]} \leq e^{\epsilon}.$$

The above inequality guarantees that an adversary cannot deduce whether two perturbed responses $A(y_1)$ and $A(y_2)$ originate from the same ($y_1 = y_2$) or different labels ($y_1 \neq y_2$) with high confidence (controlled by ϵ). As such, the adversary cannot use fine-tuned queries, no matter how many they are, to find the decision boundary within the granule of boundary-sensitive zone.

4 Boundary Differentially Private Layer

In this section, we present our solution to protect against model extraction attacks with respect to ϵ-boundary differential privacy (ϵ-BDP) by appending a BDP layer to the model output. According to Definition 2, this layer consists of two major steps — identifying query sensitivity, and perturbing the responses of sensitive queries to satisfy BDP. In what follows, we first introduce the implement of binary defense with a technique to identify sensitive queries with the notion of *corner points* and a perturbation algorithm called *boundary randomized response* to guarantee ϵ-BDP. Then we generalize it to multiclass model with a technique to identify *counter class* for developing its perturbation algorithm *multiclass boundary randomized response*. Finally, we introduce a zone-less variant with soft margin to globalize the defense while retaining majority of obfuscation inside boundary sensitive zone, where the perturbation is provided by *adaptive boundary randomized response*.

4.1 Binary Defense

We start from binary models which have only two labels — positive and negative, which are particularly popular in spam filtering, malware detection, and disease diagnosis.

4.1.1 Identifying Sensitive Queries In Binary Model

A query is identified as sensitive if it falls in the boundary-sensitive zone according to Definition 1. However, in

2. The semi-whitebox assumption is based on the fact that state-of-the-art models in specific application domains, such as image classification, are usually public knowledge. Nonetheless, our solution can also work against black-box attacks where such knowledge is proprietary.

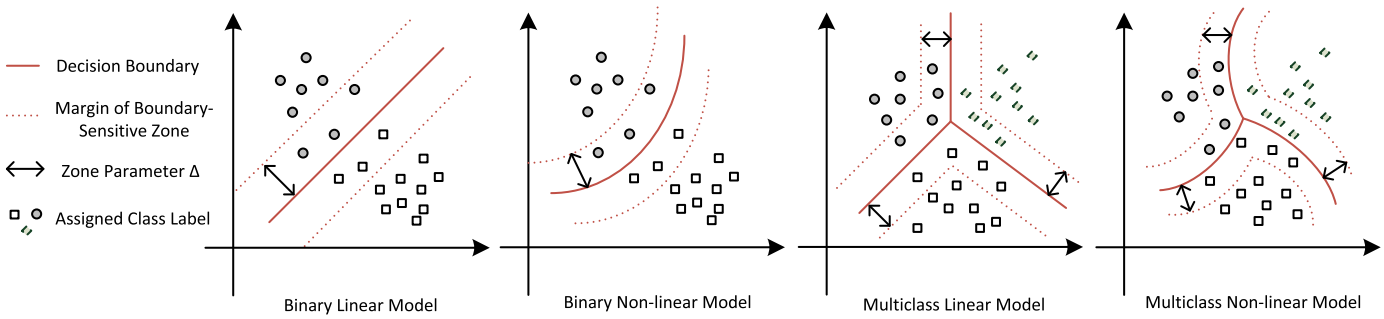


Fig. 2. Illustration of Hypothetical Decision Boundary and Boundary-Sensitive Zone in 2D.

practice the decision boundary may not have a closed form (especially for complex models such as neural networks). In this subsection, we propose a method to determine if a query $\boldsymbol{x}_q$ is sensitive without deriving the boundary-sensitive zone. The idea is to test if a ball centered at $\boldsymbol{x}_q$ with radius Δ intersects with the decision boundary.[3] In theory, this is equivalent to finding if there exists a flipping point $\boldsymbol{x}'$ in the ball that has a different label from that of the query point $\boldsymbol{x}_q$. Formally,

Definition 3 (Query Sensitivity). *A query $\boldsymbol{x}_q$ is sensitive, if and only if:*

$$\exists \boldsymbol{x}' \in B(\boldsymbol{x}_q, \Delta), s.t., f(\boldsymbol{x}') \neq f(\boldsymbol{x}_q),$$

where $B(\boldsymbol{x}_q, \Delta) = \{\boldsymbol{x} \in \mathbb{R}^d \mid dist(\boldsymbol{x}, \boldsymbol{x}_q) \leq \Delta\}$ is the ball centered at $\boldsymbol{x}_q$ with radius Δ.

The above definition needs to test infinite number of points in the ball, which is infeasible. Nonetheless, we observe that if the ball is convex and small enough,[4] a sufficient condition of query $\boldsymbol{x}_q$ being sensitive is that at least one of the *corner points* in each dimension of this ball $B(\boldsymbol{x}_q, \Delta)$ is a flipping point. As such, the sensitivity of query $\boldsymbol{x}_q$ can be approximated by testing the labels of $2d$ corner points of $\boldsymbol{x}_q$ without false negatives. Furthermore, if the distance metric is the $L1$ distance (i.e., Manhattan distance), this is also a necessary condition, which means that testing corner points leads to the exact sensivitity. For example, given a two-dimensional query $[a, b]$ and $L1$ radius Δ, if corner points $[a \pm \Delta,\ b]$ and $[a,\ b \pm \Delta]$ have flipping events, query $[a, b]$ is sensitive. The following theorem proves this.

Theorem 1 (Flipping Corner Theorem). *A sufficient condition of query $\boldsymbol{x}_q$ being sensitive is that,*

$$\exists\ \ \boldsymbol{\Delta}_i \in \Delta \cdot \boldsymbol{I},\ \ f(\boldsymbol{x}_q \pm \boldsymbol{\Delta}_i) \neq f(\boldsymbol{x}_q),$$

where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{\Delta}_i$ is the projected interval on some dimension i, and $\boldsymbol{x}_q \pm \boldsymbol{\Delta}_i$ denotes the two corner points in dimension i. If the distance metric is the $L1$ distance, this equation is also a necessary condition.

Proof. Let $\boldsymbol{x}_i$ be one of the corner points in dimension i.

- (*Sufficient Condition*) For any $\boldsymbol{x}_i$, the decision boundary must exist between $\boldsymbol{x}_i$ and $\boldsymbol{x}_q$ where $f(\boldsymbol{x}_i) \neq f(\boldsymbol{x}_q)$. It intersects line $\boldsymbol{x}_i\boldsymbol{x}_q$ at point $\boldsymbol{b}_i$. As $\boldsymbol{x}_i$, $\boldsymbol{x}_q$ and $\boldsymbol{b}_i$ are on the same straight line, we have

  $$dist(\boldsymbol{x}_i, \boldsymbol{b}_i) + dist(\boldsymbol{x}_q, \boldsymbol{b}_i) = dist(\boldsymbol{x}_i, \boldsymbol{x}_q) = \Delta.$$

  Since dist$(\boldsymbol{x}_q, f)$ is the minimum distance between $\boldsymbol{x}_q$ and any point on the decision boundary, we have

  $$dist(\boldsymbol{x}_q, f) \leq dist(\boldsymbol{x}_q, \boldsymbol{b}_i) = \Delta - dist(\boldsymbol{x}_i, \boldsymbol{b}_i) < \Delta.$$

  According to Definition 1, query $\boldsymbol{x}_q$ is sensitive and this proves the sufficient condition.
- (*Necessary Condition for $L1$ Distance*) If $\boldsymbol{x}_q$ is a sensitive query, an $L1$-ball centered at $\boldsymbol{x}_q$ with radius Δ will be given by

  $$B(\boldsymbol{x}_q, \Delta) = \{\boldsymbol{x} \in \mathbb{R}^d \mid dist_{L1}(\boldsymbol{x}, \boldsymbol{x}_q) \leq \Delta\}. \quad (1)$$

  Let $\boldsymbol{b}_m$ be the point which is the closest to $\boldsymbol{x}_q$ on the decision boundary of f. According to Definition 3, we have

  $$dist_{L1}(\boldsymbol{x}_q, \boldsymbol{b}_m) = dist_{L1}(\boldsymbol{x}_q, f) < \Delta.$$

  Since $\boldsymbol{x}_q$ is sensitive, $\boldsymbol{b}_m$ must be inside this $L1$-ball:

  $$\boldsymbol{b}_m \in B(\boldsymbol{x}_q, \Delta).$$

  This means that the decision boundary must intersect the ball at $\boldsymbol{b}_m$. As such, at least one convex vertex of the ball is on a different side of the decision boundary than point $\boldsymbol{x}_q$. Since the convex vertices of an $L1$-ball are exactly those corner points, there exists at least one corner point $\boldsymbol{x}_i$ such that $f(\boldsymbol{x}_i) \neq f(\boldsymbol{x}_q)$. And this proves the necessary condition.

Therefore, flipping corner point is a sufficient condition for query $\boldsymbol{x}_q$ being sensitive and a necessary condition under the $L1$ distance metric. □

4.1.2 Perturbation Algorithm: Boundary Randomized Response

Randomized response [11] is a privacy-preserving survey technique developed for surveying sensitive questions. A randomized boolean value is given to the answer and provides plausible deniability. As the perturbation algorithm defined in boundary differential privacy has exactly two

3. The case of tangency is rarely reached in real life given that the feature space is usually continuous. For simplicity, we mainly consider intersection.

4. If Δ is small, the decision boundary near the ball can be treated as a hyperplane.

output choices, we design the following BRR algorithm based on randomized response to satisfy ϵ-BDP.

Definition 4 (Boundary Randomized Response, BRR). *Given query sample $\boldsymbol{x}_q$ and its true response $y_q \in \{0, 1\}$, the boundary randomized response algorithm $A(y_q)$ perturbs y_q by the following:*

$$A(y_q) = \begin{cases} y_q, & \text{w.p.} \quad \frac{1}{2} + \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}} \\ 1 - y_q, & \text{w.p.} \quad \frac{1}{2} - \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}} \end{cases}.$$

Theorem 2. *The boundary randomized response algorithm $A(y_q)$ satisfies ϵ-BDP.*

Proof. See Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2020.3043382. □

4.1.3 Summary for Binary Defense

Algorithm 1 summarizes the detailed procedures of BDP layer that can be tapped to the output of any binary machine learning model f. When a new query $\boldsymbol{x}_q$ arrives, if it has already been queried before, the layer directly returns the cached response y'_q to prevent attacker from learning multiple perturbed responses of the same query response, which can lead to a less private BDP. Otherwise, the layer first obtains the real result y_q from model f. Then it determines whether $\boldsymbol{x}_q$ is in the boundary-sensitive zone by checking all corner points. As long as one corner point is as a flipping point, the query is identified as sensitive, and the boundary randomized response algorithm $BRR(\cdot)$ with privacy budget ϵ will be invoked. The layer will thus return the perturbed result y'_q and cache it for future use. Otherwise, if $\boldsymbol{x}_q$ is not sensitive after checking all corner points, the real result y_q will be returned. As for time complexity for identifying sensitive queries if they are not cached, since we only need to check two corner points in each dimension for an m-dimensional query, the upper bound time complexity will be $O(mT)$ where T is the time cost of each model prediction. Nonetheless, the average time cost can be much smaller as the process can terminate early as long as one flipping corner is found and we also propose n-shot sampling to speed up this process. As for T, in our evaluation environment we find the average of T over a variety of models is $70 - 90\ \mu s$ for logistic regression, $400 - 500\ \mu s$ for shallow neural network, and $600 - 700\ \mu s$ for convolutional neural network.

4.2 Generalization to Multiclass Model

We now consider the case where the prediction domain of a model has more than two classes. To adapt the current algorithm to multiclass model and retain ϵ-BDP guarantee, we observe that the decision boundary in a multiclass model is essentially a union of binary boundaries, each of which separates two classes. As such, we can extend the definition of sensitive query in a multiclass model in terms of its nearby decision boundary. Formally,

Definition 5 (Multiclass Query Sensitivity). *A query $\boldsymbol{x}_q$ is sensitive to the decision boundary of classes $u, v \in Domain(f)$, if and only if:*

$$\exists \boldsymbol{x}' \in B(\boldsymbol{x}_q, \Delta),\ s.t., f(\boldsymbol{x}') = u, f(\boldsymbol{x}_q) = v, u \neq v,$$

where u is called the counter class *to the true response v and $Domain(f)$ contains all possible output classes.*

In this way, one class (i.e., true response) can be treated as the "positive" label and the other as the "negative" one (i.e., the counter class). The multiclass case is thus reduced to the same problem of protecting binary decision boundaries except that there are boundaries for each pair of classes.

Algorithm 1. Boundary Differentially Private Layer For Binary Model

Input: Boundary-Sensitive Zone Parameter Δ
Boundary Privacy Budget ϵ
Query $\boldsymbol{x}_q \in R^d$
Model f
Output: Original Response y_q or Perturbed Response y'_q
Procedure:

```
if x_q is not cached then
    y_q = f(x_q)
    CornerPoints = getCornerPoints(Δ, x_q)
    for x_i in CornerPoints do
        if x_i is a flipping point then
            y'_q = BRR(y_q, ε)
            Cache(x_q, y'_q)
            return y'_q
    return y_q
else
    return getCached(x_q)
```

4.2.1 Identifying Counter Class

The key idea of avoiding multiple decision boundaries from a variety of candidate counter classes for u is to only associate sensitive query with its nearest decision boundary and identify the corresponding class on the other side as the counter class. To identify this class, we use the majority vote from all flipping corner points. However, a full scan of all these points is not practical particularly in a high dimensional dataset with hundreds or even thousands of corner points. To strike a balance between accuracy and efficiency, we perform an N-shot sampling over the flipping corners. Formally,

Definition 6 (N-shot Flipping Corner). *An estimate of query $\boldsymbol{x}_q$ being sensitive to the decision boundary of classes $u, v \in Domain(f)$ is that,*

$$\forall\ \boldsymbol{\Delta}_{i \in N} \in \Delta \cdot \boldsymbol{I},\ V(f(\boldsymbol{x}_q \pm \boldsymbol{\Delta}_i) \neq f(\boldsymbol{x}_q)) = u,$$

where $\boldsymbol{\Delta}_{i \in N}$ denotes flipping corner points in N sampling dimensions with flipping corners and $V(\cdot)$ finds the class with the highest flipping rate from the comparison results of provided corner points.

4.2.2 Multiclass Boundary Randomize Response

Now that we can detect a sensitive query in the multiclass case, given its true response and counter class, we use the following Multiclass Boundary Randomized Response (MBRR) algorithm to achieve ϵ-BDP.

Definition 7 (Multiclass Boundary Randomized Response, MBRR). *Given a query sample $\boldsymbol{x}_q$, its true response y_q and counter class $\overline{y_q}$ (y_q, $\overline{y_q} \in \{u, v\}$), the multiclass boundary randomized response algorithm $A(y_q)$ perturbs y_q by the following:*

$$A(y_q) = \begin{cases} y_q, & \text{w.p.} \quad \frac{1}{2} + \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}} \\ \overline{y_q}, & \text{w.p.} \quad \frac{1}{2} - \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}} \end{cases}.$$

Theorem 3. *The multiclass boundary randomized response algorithm $A(y_q)$ provides ϵ-BDP to each binary decision boundary $f_{u,v}$ in the model.*

The proof is similar to that of Theorem 2 and is thus omitted.

4.2.3 Summary for Multiclass Defense

Algorithm 2 summarizes the detailed procedures of BDP layer for a multiclass machine learning model f. Similar to the binary model, the layer directly returns a cached response to retain privacy guarantee if query $\boldsymbol{x}_q$ has been processed before. In addition to zone parameter Δ and privacy budget ϵ, the number of samples N to determine the counter class can be tuned between efficiency and accuracy. After the counter class is determined, the multiclass boundary randomized response algorithm $MBRR(\cdot)$ with privacy budget ϵ is invoked. The layer then returns the perturbed result y'_q and caches it for future use.

Algorithm 2. Boundary Differentially Private Layer For Multiclass Model

Input: Flipping Corner Shot N
Boundary-Sensitive Zone Parameter Δ
Boundary Privacy Budget ϵ
Query $\boldsymbol{x}_q \in R^d$
Model f
Output: Original Response y_q or Perturbed Response y'_q
Procedure:

```
if x_q is not cached then
    y_q = f(x_q)
    cPoints = getCornerPoints(Δ, x_q, N)
    for x_i in CornerPoints do
        if x_i is a flipping point then
            CounterClass = getCounterClass(cPoints)
            y'_q = MBRR(y_q, ε, CounterClass)
            Cache(x_q, y'_q)
            return y'_q
    return y_q
else
    return getCached(x_q)
```

4.3 Zone-less Boundary Differentially Private Layer

In the previous section, the decision boundary is formulated as a zone with a hard margin controlled by Δ — a query is either inside this zone (i.e., sensitive) or outside of it (i.e., non-sensitive). ϵ-BDP can be achieved in the former case but no privacy is provided in the latter case. This makes the choice of Δ a crucial and challenging task for the user — a small value leaves some decision boundary unprotected and yet a large value introduces unnecessary noise to non-boundary area where no protection is needed. To make Δ less influential, in this section we propose a soft margin approach as an alternative to the hard margin.

4.3.1 Soft Query Sensitivity

The soft margin is essentially defined through the notion of soft query sensitivity, in which a query is no longer a hard "0" (non-sensitive) or "1" (sensitive). Instead, it is 1 on the soft margin and is larger than 1 when inside the margin. Then the degree of perturbation depends on the query sensitivity. The rationale behind a soft sensitivity of a query is three-folded. First, it must have a negative correlation with its distance to the nearest decision boundary, because query results reveal more information about the boundary and thus are more sensitive when they are closer to it. Second, how much the sensitivity depends on the distance should be controlled by the model owner. Third, the soft and hard sensitivity should be a unified notion. That is, the perturbation protocol for the hard sensitivity, Boundary Randomized Response (BRR), must still work with minimum adaptation. The following is a definition that satisfies all three rationales.

Definition 8 (Soft Query Sensitivity). *Given a model f and a zone parameter Δ chosen by the model owner, the sensitivity of a query $\boldsymbol{x_q}$ is a fractional function as:*

$$s(\boldsymbol{x_q}) = \frac{\Delta}{dist(\boldsymbol{x_q}, f_{u,v})}, \tag{2}$$

where $dist(\boldsymbol{x_q}, f_{u,v})$ measures the distance between query $\boldsymbol{x_q}$ and the nearest decision boundary $f_{u,v}$.

To derive the distance without a closed form of the decision boundary, we can still adopt the flipping-corner-point method. The idea is to perform a binary search with an initial distance guess. If flipping corner points occur, the distance must be smaller, so we reduce the current guess to half and repeat the search; otherwise we double the guess. The final distance is obtained when a precision threshold or a maximum number of iterations is reached.

4.3.2 Adaptive Boundary Randomized Response

Given the above definition of query sensitivity, the perturbation algorithm, Adaptive Boundary Randomized Response (ABRR), is exactly the same as BRR, except for the exponents. In BRR, the exponent is ϵ which is implicitly $\frac{\epsilon}{s(\boldsymbol{x_q})}$ where $s(\boldsymbol{x_q})$ is always 1. ABRR uses the same formulae where $s(\boldsymbol{x_q})$ is defined in Eq. (2).

Definition 9 (Adaptive Boundary Randomized Response, ABRR). *Given query sample $\boldsymbol{x}_q$ normalized to [-1,1], query sensitivity $s(\boldsymbol{x_q})$, its true response y_q and counter class $\overline{y_q}$ (y_q, $\overline{y_q} \in \{u, v\}$), the adaptive boundary randomized response algorithm $A(y_q)$ perturbs y_q by the following:*

$$A(y_q) = \begin{cases} y_q, & \text{w.p.} \quad \frac{1}{2} + \frac{\sqrt{e^{2\psi}-1}}{2+2e^{\psi}} \\ \overline{y_q}, & \text{w.p.} \quad \frac{1}{2} - \frac{\sqrt{e^{2\psi}-1}}{2+2e^{\psi}} \end{cases},$$

where $\psi = \frac{\epsilon}{s(\boldsymbol{x_q})}$.

Theorem 4. *The adaptive boundary randomized response algorithm $A(y_q)$ satisfies ψ-BDP to each binary decision boundary $f_{u,v}$, where $\psi \leq \frac{2}{\Delta}\epsilon$.*

Proof. We assume p_1, p_2 are the probabilities of retaining true responses for two queries $\boldsymbol{x}_1, \boldsymbol{x}_2$. According to ABRR, for any two responses $y_1, y_2 \in \{u, v\}$, the four possible cases to derive the BDP inequality are:

$$\frac{Pr[y_1 = y_2 | A(y_1) = u, A(y_2) = u]}{Pr[y_1 \neq y_2 | A(y_1) = u, A(y_2) = u]}, or$$
$$\frac{Pr[y_1 = y_2 | A(y_1) = v, A(y_2) = v]}{Pr[y_1 \neq y_2 | A(y_1) = v, A(y_2) = v]} = \frac{p_1 p_2 + (1-p_1)(1-p_2)}{p_1(1-p_2) + p_2(1-p_1)},$$

and

$$\frac{Pr[y_1 = y_2 | A(y_1) = u, A(y_2) = v]}{Pr[y_1 \neq y_2 | A(y_1) = u, A(y_2) = v]}, or$$
$$\frac{Pr[y_1 = y_2 | A(y_1) = v, A(y_2) = u]}{Pr[y_1 \neq y_2 | A(y_1) = v, A(y_2) = u]} = \frac{p_1(1-p_2) + p_2(1-p_1)}{p_1 p_2 + (1-p_1)(1-p_2)}.$$

Since $p_1, p_2 \in [\frac{1}{2}, 1)$, the partial derivatives to p_1 and p_2 of the right-hand side term in the former two cases are

$$\frac{\partial \frac{p_1 p_2 + (1-p_1)(1-p_2)}{p_1(1-p_2) + p_2(1-p_1)}}{\partial p_1} = \frac{2p_1 - 1}{(-2p_1 p_2 + p_1 + p_2)^2} \geq 0,$$

$$\frac{\partial \frac{p_1 p_2 + (1-p_1)(1-p_2)}{p_1(1-p_2) + p_2(1-p_1)}}{\partial p_2} = \frac{2p_2 - 1}{(-2p_1 p_2 + p_1 + p_2)^2} \geq 0.$$

As such, the right-hand side term in the former two cases are monotonically increasing. Similarly, that term in the latter two cases are monotonically decreasing. Let $p_{max} = \max\{p_1, p_2\}$. Then the two terms are bounded as follows.

$$\frac{p_1 p_2 + (1-p_1)(1-p_2)}{p_1(1-p_2) + p_2(1-p_1)} \leq \frac{{p_{max}}^2 + (1-p_{max})^2}{2p_{max}(1-p_{max})}, \tag{3}$$

$$\frac{p_1(1-p_2) + p_2(1-p_1)}{p_1 p_2 + (1-p_1)(1-p_2)} \leq 1. \tag{4}$$

Furthermore, since $p_1, p_2 \in [\frac{1}{2}, 1)$, the right-hand side term of Eq. (3) also serves as the upper bound of the right-hand side term of Eq. (4). That is,

$$\frac{{p_{max}}^2 + (1-p_{max})^2}{2p_{max}(1-p_{max})} \geq 1.$$

According to ABRR, we can derive p_{max} as

$$p_{max} = \frac{\sqrt{e^{2\psi_{max}} - 1}}{2 + 2e^{\psi_{max}}}.$$

By replacing p_{max} in the right term of Eq. (3) with it, we have

$$\frac{{p_{max}}^2 + (1-p_{max})^2}{2p_{max}(1-p_{max})} = e^{\psi_{max}}.$$

Finally, since the sensitivity $s(\boldsymbol{x_q})$ is in the range $[\frac{2}{\Delta}, +\infty]$, we derive the bound of ψ_{max} as

$$\psi_{max} \leq \frac{\epsilon}{s(\boldsymbol{x_q})} \leq \frac{\Delta}{2}\epsilon. \tag{5}$$

Due to the monotonicity of exponential function, we have

$$e^{\psi_{max}} \leq e^{\frac{2}{\Delta}\epsilon}. \tag{6}$$

Therefore, for any two queries, the algorithm satisfies ψ-BDP where $\psi \leq \frac{2}{\Delta}\epsilon$. □

Algorithm 3. Zone-Less Boundary Differentially Private Layer for Multiclass Model

Input: Query $\boldsymbol{x}_q \in R^d$
Model f
Soft Margin Δ
Boundary Privacy Budget ϵ
Output: Perturbed Response y'_q
Procedure:
if $\boldsymbol{x}_q$ is not cached **then**
  $y_q = f(\boldsymbol{x}_q)$
  $s_q = getSensitivity(f, \boldsymbol{x}_q, \Delta)$
  $CounterClass = getCounterClass(f, \boldsymbol{x}_q)$
  $y'_q = ABRR(y_q, \epsilon, s_q, CounterClass)$
  $Cache(\boldsymbol{x}_q, y'_q)$
  return y'_q
else
  return $getCached(\boldsymbol{x}_q)$

Notably, for queries inside the margin of Δ, sensitivity is equal or greater than 1. As a result, we can prove that a minimum of ϵ-BDP is always achieved, same as the requirement for boundary-sensitive zone in BDPL. In other words, ABRR essentially provides stronger non-uniform ϵ-BDP than BRR in Δ boundary-sensitive zone. We summarize this property as follows.

Corollary 4.1. *For any query $\boldsymbol{x}_q$ inside the margin of Δ, the adaptive boundary randomized response algorithm $A(y_q)$ satisfies ψ-BDP where $\psi \leq \epsilon$*

Proof. Since query $\boldsymbol{x}_q$ now has $s(\boldsymbol{x}_q)$ in the range $[1, +\infty]$, , we can prove the following by Eq. (5) and (6).

$$e^{\psi_{max}} \leq e^{\epsilon}$$ □

4.3.3 Summary for Zone-Less Defense

Algorithm 3 summarizes the detailed procedures of zone-less BDP layer with soft margin. Caching policy is still carried out for any historical query. If $\boldsymbol{x}_q$ is a new query, the sensitivity of $\boldsymbol{x}_q$ to nearest decision boundary is first measured using a binary search with the corner-point technique. Then the counter class is calculated for a multiclass model. Finally, the adaptive boundary randomized response algorithm $ABRR(\cdot)$ is invoked to perform perturbation with BDP protection.

5 Experiments

In this section, we evaluate the effectiveness of boundary differentially private layer (BDPL) against model extraction attacks. Specifically, we implement those motivating extraction attacks using fine-tuned queries as in [1], [5] and compare the success rates of these attacks with and without BDPL.

5.1 Setup

5.1.1 Datasets and Machine Learning Models

We evaluate three datasets and two models used in the literature [1] — a Botany dataset *Mushrooms* (113 attributes, 8124 records), a census dataset *Adult* (109 attributes, 48842 records) and a general social survey dataset *GSS* (101 attributes, 16127 records). The former two datasets are obtained from UCI machine learning repository [12] while the last one is from NORC [13]. All categorical items are processed by *one-hot-encoding* [14] and missing values are replaced with the mean value of this attribute. We adopt *min-max normalization* to unify all feature domains into [-1,1]. Data augmentation is not used for all the experiments, in accordance with the configuration of the original attacks.

For the evaluation of binary defense, *Mushrooms* dataset is trained on the label that shows whether a mushroom is poisonous or edible, and *Adult* dataset is trained on the label that shows whether the annual income of an adult exceeds 50K (Adult-b). As for multiclass defense, *GSS* dataset is used to train a model to predict level of happiness while *Adult* dataset is used to predict the race of the participants (Adult-m).

We train both a linear model, namely, logistic regression (LR), and a non-linear model, namely, 3-layer neural network (NN), to predict unknown labels on the above datasets. Logistic regression is implemented using *cross-entropy* loss with $L2$ regularizer. Neural network is implemented using TensorFlow r1.12 [15]. The hidden layer contains 20 neurons with $tanh$ activation. The output layer is implemented with a $sigmoid$ function for binary prediction and a $softmax$ function for multiclass prediction.

5.1.2 Attack and Evaluation Metrics

We implement the extraction attack defined in Section 2 using original attack code of line-search technique in [1]. Specifically, the attacker first creates a seed set of pairwise queries with opposite or different response labels, and then searches for new samples that lie on the line segment connecting this pair to approach the decision boundary. This process is repeated until either a searching threshold or query limit is reached. The size of a seed set is 4 and the searching threshold is 0.05. It is a white-box attack which produces an extracted model f' with the same hyperparameters and architectures as the original model f. To compare f and f', we adopt *extraction rate* [1], [6] to measure the proportion of matching predictions (i.e., both f and f' predict the same label) in an evaluation query set. Formally,

- *Extraction Rate (R)*. Given an evaluation query set $\mathcal{X}_e$, the extraction rate

$$R = \frac{1}{|\mathcal{X}_e|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_e} \mathbb{1}(f(\boldsymbol{x}_i) = f'(\boldsymbol{x}_i)),$$

  where $\mathbb{1}(\cdot)$ is an indicator function that outputs 1 if the input condition holds and 0 otherwise. The extraction rate essentially measures the similarity of model outputs given the same inputs.
- *Utility (U)*. This second metric is evaluated on the test data points and measures the proportion of responses that are perturbed (i.e., flipped) by BDPL. It indicates how useful these responses are from a normal user's perspective. Formally, given the entire queries $\mathcal{X}_q$ issued from test set by clients, and the set of (perturbed) responses $\mathcal{Y}_q$ from the service provider,

$$U = \frac{1}{|\mathcal{X}_q|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_q, y_i \in \mathcal{Y}_q} \mathbb{1}(f(\boldsymbol{x}_i) = y_i).$$

We follow the same evaluation setting as the original works. In the classic attack (Tramer's [1]) of Sections 5.2, 5.3, 5.4, and 5.5, training set sample are used for the construction of victim model, whereas test set samples and uniformly sampled points are applied in the evaluation of utility and extraction rate respectively. In the advanced attack (ActiveThief [8]) of Section 5.6, the configuration is similar, except that the extraction rate is evaluated against test set samples.

5.2 Overall Evaluation

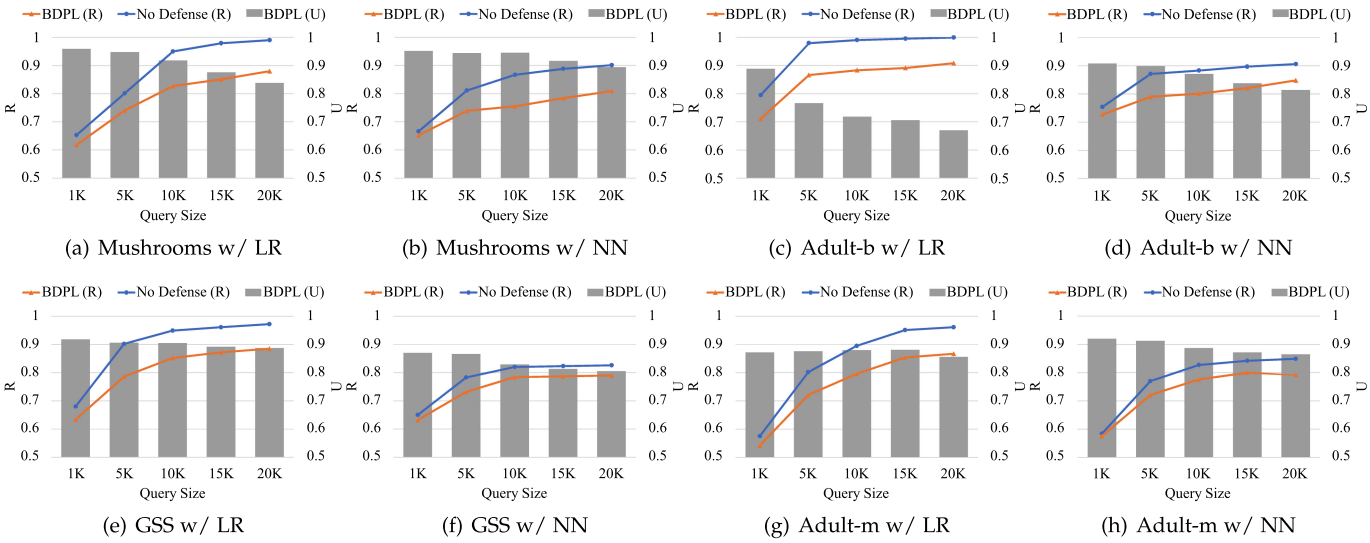
To evaluate how well the decision boundary can be protected by our solution, we launch extraction attacks on a number of model/dataset combinations and plot the extraction rate R of sensitive queries in Figs. 3 and 4 in terms of the number of queries.

Evaluation of BDPL. In this experiment, we set $\Delta = 1/8$, and $\epsilon = 0.01$ for all models. As shown in Fig. 3, except for the initial extraction stage (query size less than $5K$), BDPL exhibits a significant protection effect for all 8 combinations — up to 12 percent drop on R — compared with no defense. The drops in *GSS w/ NN* and *Adult-m w/ NN* are smaller (around $5\% - 6\%$) because these two models are the most complicated (multiclass neural networks) and the least vulnerable to label perturbation.

The secondary axis of Fig. 3 also plots the utility of BDPL using bar chart. We observe that the utility saturates at over 80 percent after $20K$ queries in all combinations (among which 4 can achieve nearly 90 percent utility) except for *Adult w/ LR*. This model has the fewest parameters and feature inputs, so BDPL has to perturb more sensitive queries to retain the same BDP level as the others. The impact on utility by Δ and ϵ will be further discussed in Section 5.4.

It is noteworthy that 1 percent reduction of extraction rate is more significant in later attack stage than earlier stage where the attackers need to increase the amounts of queries tremendously. For example, in Fig. 6e, the adversary spends $15K$ queries to improve extraction rate from 90 to 97 percent, which is canceled off by BDPL using after $15K$ queries with only 11 percent utility loss. Obviously, 7 percent drop on extraction rate is more significant than an 11 percent utility loss because the former costs 15K queries whereas the latter costs only about 1.5K queries.

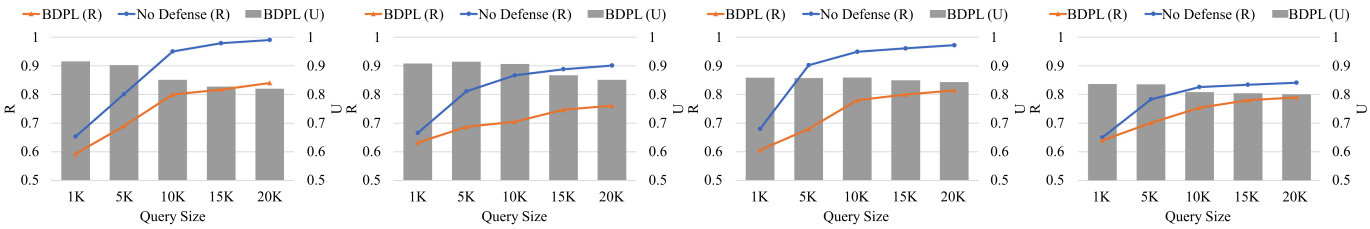
Evaluation of Zone-Less BDPL. In this experiment, we set $\Delta = 1/8$ and increase ϵ to 0.16 so that the overall utility is similar to the previous experiment (i.e., over 80 percent). The experiment results on Mushrooms and GSS are plotted in Fig. 4. Zone-less BDPL provides even better protection in all 4 combinations than BDPL, particularly at the initial stage (query size less than $5K$) with a much lower extraction rate. Furthermore, all extraction rates saturate even earlier (after $10k$ of queries) than BDPL. Overall, we observe that zone-less BDPL performs particularly well with logistic



(a) Mushrooms w/ LR (b) Mushrooms w/ NN (c) Adult-b w/ LR (d) Adult-b w/ NN

(e) GSS w/ LR (f) GSS w/ NN (g) Adult-m w/ LR (h) Adult-m w/ NN

Fig. 3. Overall protection effect by BDPL: Extraction rate and utility.



(a) Mushrooms w/ LR (b) Mushrooms w/ NN (c) GSS w/ LR (d) GSS w/ NN

Fig. 4. Overall protection effect by zone-less BDPL: Extraction rate and utility.

regression models, where we witness an extra drop of 4 percent on R compared with BDPL. The impact on R and U by Δ and ϵ will be shown in Section 5.5.

5.3 BDPL Versus Uniform Perturbation

In this experiment, we compare BDPL on binary model (single decision boundary) with a uniform perturbation mechanism that randomly flips the response label by a certain probability, whether the query is sensitive or not. To have a fair comparison, we use trial-and-error[5] to find this probability so that the overall extraction rates of both mechanisms are almost the same. In Fig. 5, we plot the extraction rates of both mechanisms for *Mushrooms w/ LR* with $\Delta = 1/8$ and $\epsilon = 0.01$. We observe that BDPL outperforms uniform perturbation by $5\% - 7\%$ extraction rate, which is very significant as this leads to an increase of misclassification rate by $30\% - 50\%$. As such, we can conclude that BDPL is very effective in protecting the decision boundary by differentiating sensitive queries from non-sensitive ones, and therefore it retains high utility for query samples that are faraway from the boundary.

5.4 Impact of ϵ and Δ in BDPL

In this subsection, we evaluate BDPL performance with respect to zone parameter Δ and privacy budget ϵ. In Fig. 6, we fix ϵ and vary Δ from $1/64$ to $1/8$ for all 8 model/dataset combinations. In Fig. 7, we fix Δ and vary ϵ from 0.01 to 0.64 for all 8 model/dataset combinations.

Impact on Extraction Rate. When Δ increases from $1/64$ to $1/8$, the extraction rate is significantly reduced in both logistic regression (up to 12 percent drop) and neural network (up to 10 percent drop). Nonetheless, for neural networks, the extract rate does not change much when Δ increases from $1/64$ to $1/32$, which indicates that if the boundary-sensitive zone is too small, BDPL may not provide effective protection, especially when the decision boundary is non-linear. As for privacy budget ϵ, its impact is not as significant as Δ. We only observe up to 4 percent drop of extraction rate when ϵ decreases from 0.64 to 0.01 for all 8 model/dataset combinations.

Last but not the least, the extraction rates under all these settings saturate as the query size increases. In most cases, they start to saturate before $5K$ queries, and in the worst case, they saturate at $15K$ or $20K$. This indicates that BDPL imposes a theoretical upper bound on the extraction rate no matter how many queries are issued.

Impact on Utility. In this part, we evaluate BDPL performance regarding utility under similar varying settings. In Fig. 8, we plot the final utility with respect to Δ and ϵ after $20K$ queries for all model/dataset combinations. Except for *Adult w/ LR*, all utilities are higher than 80 percent and most of them are above 90 percent, which means that BDPL does not severely sacrifice the accuracy of a machine learning service. As expected, the utility reaches peak when $\Delta = 1/64$ (the smallest zone size) and $\epsilon = 0.64$ (the least probability of

5. To do this, we start with 1 random flip out of all responses and measure its overall extraction rate. We then repeatedly increment this number by 1 until the overall extraction rate is very close to that of BDPL.

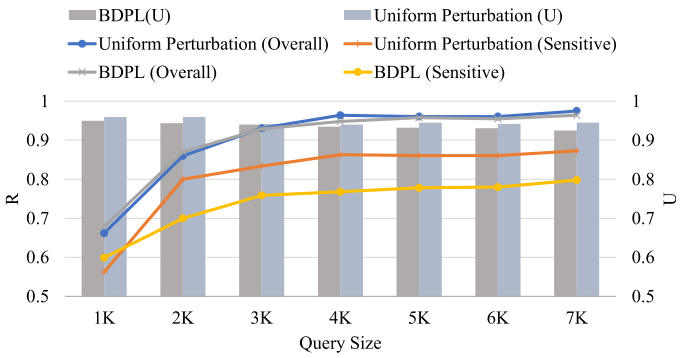


Fig. 5. BDPL versus uniform perturbation.

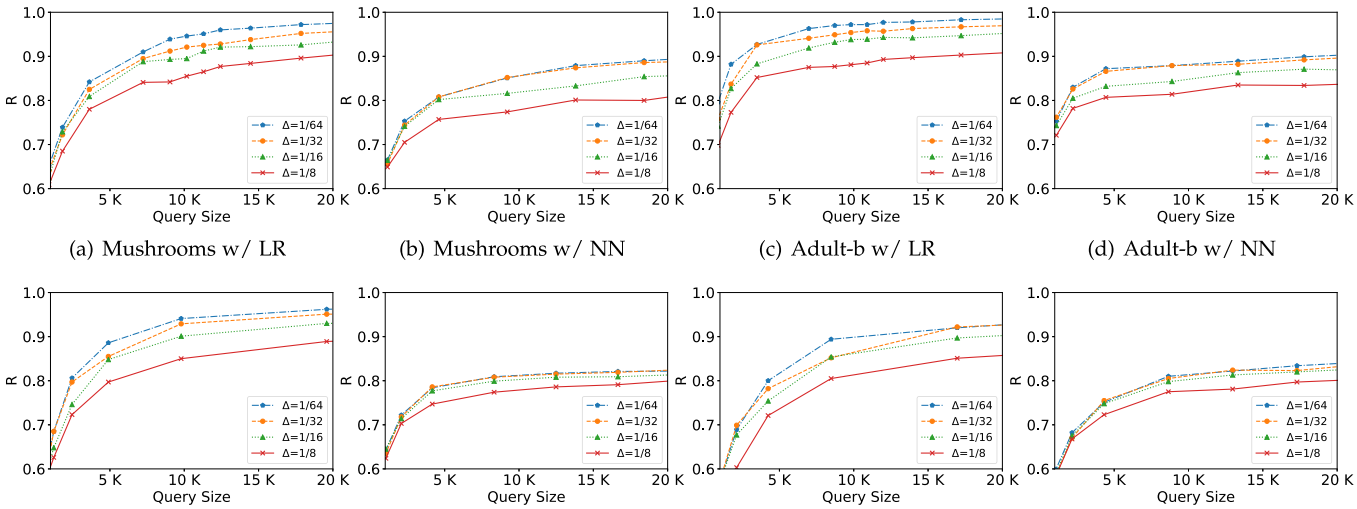
perturbation). Furthermore, as is coincided with the extraction rate, the utility is more sensitive to Δ than to ϵ. For example, an increase of Δ from 0.01 to 0.1 leads to a drop of utility by 10 percent, whereas a decrease of ϵ from 0.1 to 0.01 leads to only 5 percent drop.

To conclude, BDPL permanently protects decision boundary of both linear and non-linear models with moderate utility loss. The changes of Δ and ϵ (particularly the former) have modest impact on the extraction rate and utility.

5.5 Impact of ϵ and Δ in Zone-Less BDPL

In this subsection, we evaluate zone-less BDPL performance with respect to Δ and ϵ. In Fig. 9, we fix ϵ and vary Δ from $1/64$ to $1/8$. In Fig. 10, we fix Δ and vary ϵ from 0.01 to 0.64. Due to space limitation, we only plot the results on 2 dataset/model combinations, i.e., *GSS w/ LR* and *GSS w/ NN*.

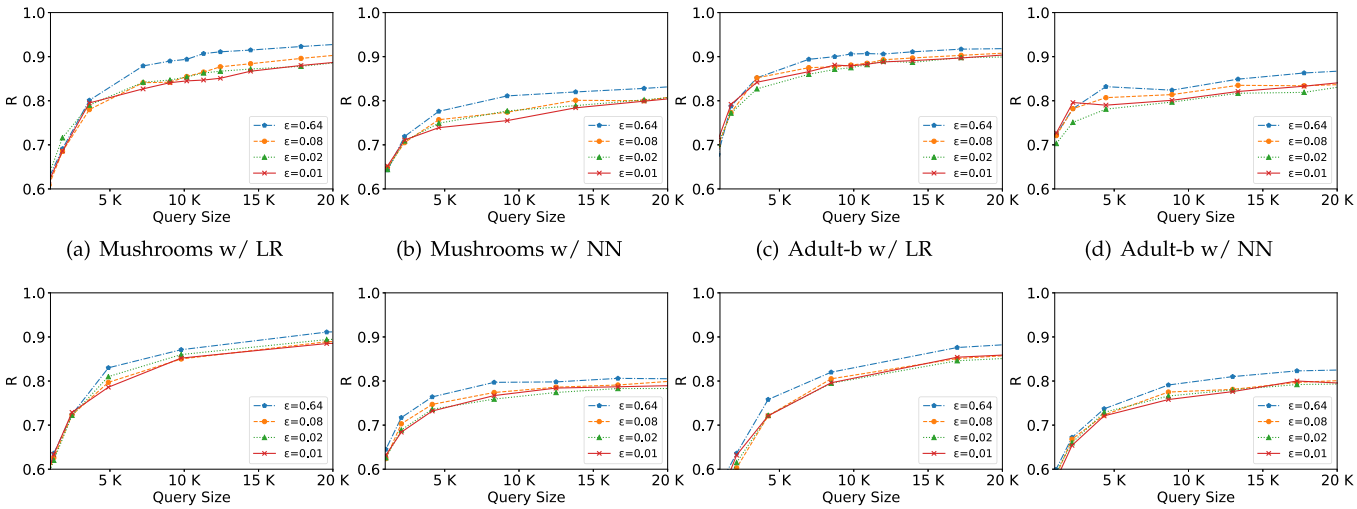
Impact on Extraction Rate. Both parameters maintain effectiveness in protecting decision boundary and saturating the extraction rate. Particularly, compared to the hard margin solution, when ϵ decreases from 0.64 to 0.01, zone-less BDPL draws significant drop over extraction rate (up to 25 percent drop in logistic regression and 15 percent in neural network). This coincides with Corollary 4.1 in Section 4.3 that zone-less BDPL achieves better ϵ-BDP protection than BDPL. Meanwhile, varying zone parameter Δ has less eminent effect than in the hard margin case. This coincides with our zone-less design to protect the decision boundary with a soft margin.



(a) Mushrooms w/ LR (b) Mushrooms w/ NN (c) Adult-b w/ LR (d) Adult-b w/ NN

(e) GSS w/ LR (f) GSS w/ NN (g) Adult-m w/ LR (h) Adult-m w/ NN

Fig. 6. Impact of Varying Δ in BDPL with $\epsilon = 0.01$.



(a) Mushrooms w/ LR (b) Mushrooms w/ NN (c) Adult-b w/ LR (d) Adult-b w/ NN

(e) GSS w/ LR (f) GSS w/ NN (g) Adult-m w/ LR (h) Adult-m w/ NN

Fig. 7. Impact of varying ϵ in BDPL with $\Delta = 1/8$.

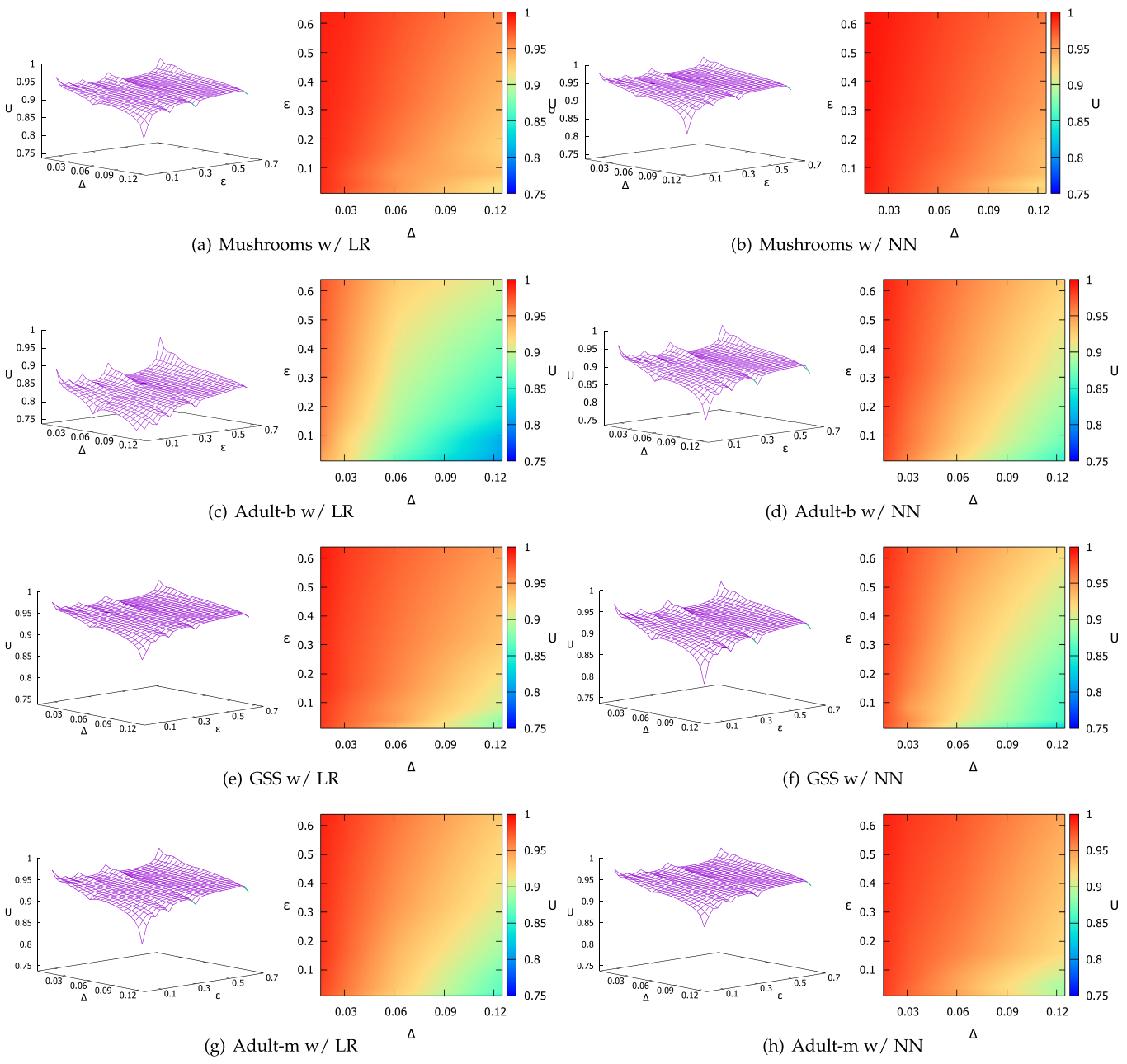


Fig. 8. Utility versus Δ and ϵ in BDPL.

Impact on Utility. We evaluate zone-less BDPL in terms of utility after $20K$ queries. In Fig. 11, we observe that the change of ϵ leads to 35 percent change of utility while the change of Δ only leads to 10 percent. This can be explained by the fact that zone-less BDPL adopts ABRR which obfuscates results in the global feature space . In addition, utility is still independent of model types and remains over 80 percent when ϵ is greater than 0.15. As expected, utility reaches peak when $\Delta = 1/64$ (when soft margin is the most concentrated near a decision boundary) and $\epsilon = 0.64$ (the least probability of perturbation).

To conclude, zone-less BDPL provides strong protection for decision boundary in the global feature space. Privacy budget ϵ brings more control over the extraction rate than Δ.

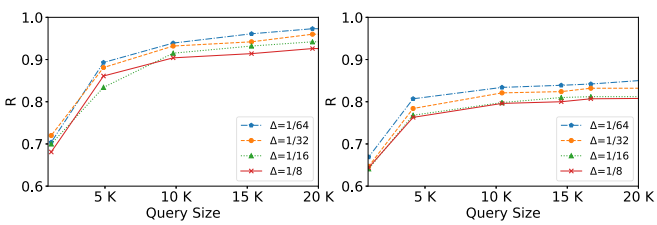
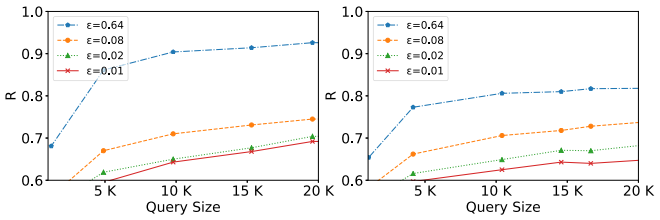


Fig. 9. Impact of varying Δ in zone-less BDPL.

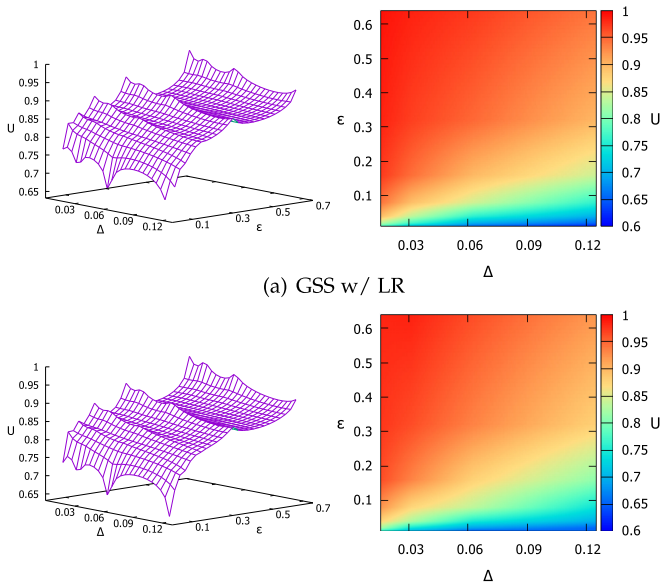
5.6 Evaluation of BDPL on Advanced Attack

Recent study has shown that the extraction attacks are becoming threatening on complex models such as convolutional neural network. In this subsection, we turn to these emerging attacks which substantially scale both the input dimensions and model complexity. We expect these to be bigger challenges for BDPL as these attacks allow attackers



(a) GSS w/ LR (b) GSS w/ NN

Fig. 10. Impact of varying ϵ in zone-less BDPL.



(a) GSS w/ LR

Fig. 11. Utility versus Δ and ϵ in zone-less BDPL.

to draw natural data as query from the same domain such as images.

In Table 1, we review those high-quality extraction attacks on complex model from peer-reviewed papers. To precisely address the feature of recent attacks, we list out whether the attacks support two of the most predominant advanced models, i.e., convolutional neural network (CNN) and recurrent neural network (RNN). Adversary knowledge is leveraged to illustrate adversary capability on data acquisition, specifically whether they can access any problem domain dataset. They are divided into three levels with an increasingly stringent requirement on dataset knowledge. We also categorize query strategy based on the nature of query such as reinforce-based probing (e.g., reinforce learning) and adversarial-based probing (e.g., adversarial samples). Detailed techniques can be found in the related works.

Due to space limitation, we select state-of-the-art ActiveThief [8] as the attack scheme for evaluation because it is the most recent and advanced attack. Activethief is a model extraction framework for neural networks using non-problem domain datasets and pool-based active learning strategies. We adopt the same configuration of the original paper and implement it on a convolutional neural network with various image datasets.

Datasets and Machine Learning Models. We evaluate two datasets for training victim models — a hand-written digits image dataset *MNIST* ($28 * 28$ resolution, 1 channel, $60k$ records) and a colorful general objects dataset *CIFAR10* ($32 * 32$ resolution, 3 channels, $60k$ records). The two datasets are obtained from their official repository respectively [17][18]. Compared to previous evaluation, feature size has scaled to 7x and 30x respectively. As for adversary query database, we use the downsampled and unannotated subset of the ILSVRC2012-14 dataset from ImageNet [19]. In each experiment, images from ImageNet are resized to fit the input size of the victim model.

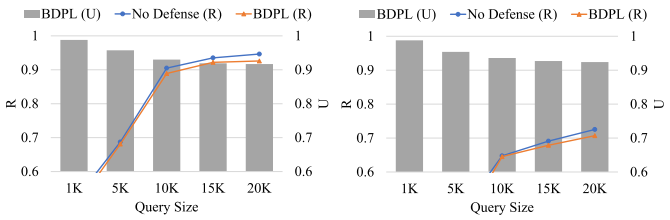
With regards to the model architecture, we adopt the same CNN design in [8] for evaluation, which has 3 blocks of convolution. In each block, there are 2 repeated units of 2 convolution layers using a 3×3 kernel, followed by 1 pooling layer using 2×2 kernel. The stride length is 1 and 2 for convolution kernel and pooling kernel respectively. ReLU is adopted as the activation function for each convolution layer. The last pooling layer is attached to a fully connected layer which produces a final prediction using softmax function.

Attack and Evaluation Metrics. The attack is performed as follows. A random subset of initial seed images are selected from the adversary database. Then the attacker queries these images against the victim model and obtains a set of responses. A basic replica model is developed by training on these query-response pairs. The attacker then queries the remaining images using a designated strategy.

We evaluate one non-probing and one probing strategy regarding decision boundary. The non-probing strategy is ActiveThief Random Strategy (ATRS) where a subset of images are selected uniformly at random. The probing strategy is ActiveThief Hybrid Strategy (ATHS) where k-center

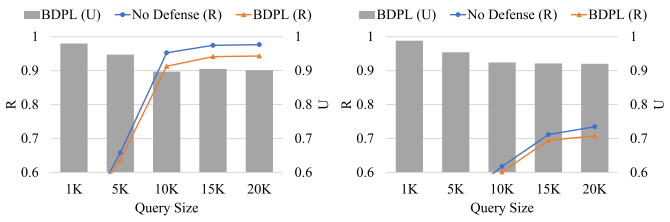
TABLE 1
Recent Advances in Model Extraction Attacks

Features		Attacks				
		Papernot *et al.* [3]	Juuti *et al.* [7]	Orekondy *et al.* [9]	Yu *et al.* [16]	Pal *et al.* [8]
Victim Model	CNN	✓	✓	✓	✓	✓
	RNN					✓
Adversary Knowledge	Problem Data	✓	✓			
	Non-problem Data			✓		
	Non-problem Data without Labels				✓	✓
Main Query Strategy	Passive Query			✓	✓	✓
	Reinforce-based Probing			✓		
	Adversarial-based Probing	✓	✓		✓	✓
API Output Level	Probability		✓	✓	✓	✓
	Label	✓	✓		✓	✓



(a) MNIST w/ CNN (b) CIFAR-10 w/ CNN

Fig. 12. Overall protection effect by BDPL on ATRS: Extraction rate and utility.



(a) MNIST w/ CNN (b) CIFAR-10 w/ CNN

Fig. 13. Overall protection effect by BDPL on ATHS: Extraction rate and utility.

and DeeplFool are combined for subset selection and it is the strongest attack in ActiveThief. This process is repeated for a fixed number of iterations. In each iteration, the replica model is retrained from all accumulated query-response pairs. Strategy hyperparameters such as number of seed samples and iteration numbers are the same in [8]. Previous evaluation metrics are adopted, that is, extraction rate R and utility U. To be consistent with original attack, the extraction rate is evaluated against test set samples in the following experiments.

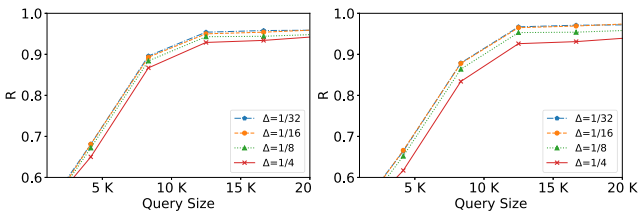
5.6.1 *Effectiveness of BDPL on Advanced Attack*

We launch two extraction attacks (ATRS and ATHS) on 2 models and plot the extraction rate R in Figs. 12 and 13 in terms of the budget of queries. BDPL parameters $\Delta = 1/7$ and $\epsilon = 0.01$ are set for all models in this experiment.

*Effectiveness on ATRS.*Fig. 12 presents the evaluation results on the non-probing attack. Despite the significant growth of attack complexity, our defense still draws a 1.5 percent drop over extraction rate on both models. The decrease is small because BDPL is focused on queries neighboring decision boundary whereas ATRS draws queries uniformly from normal image distribution and ratios of sensitive queries may be low. The mismatch leads to smaller perturbation as expected. Nevertheless, BDPL still maintains a decreased and saturated upper bound for model extraction.

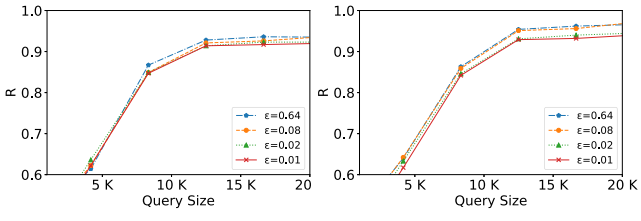
From the secondary axis of Fig. 12, we observe that the utility trend stabilizes at over 90 percent after $20K$ queries for both models. Particularly, the utility remains over 95 percent before $10K$ for *CIFAR-10 w/ CNN*. This is consistent with the small drop of extraction rate given that perturbation is light. The impact on utility by Δ and ϵ will be further discussed in Section 5.6.2.

*Effectiveness on ATHS.*Fig. 13 presents the evaluation results on the probing attack. Notably, the extraction is reduced by 4 and 2.7 percent respectively for two models, which is more significant compared to that in ATRS. This



(a) ATRS (b) ATHS

Fig. 14. Impact of varying Δ in BDPL on MNIST w/ CNN.



(a) ATRS (b) ATHS

Fig. 15. Impact of varying ϵ in BDPL on MNIST w/ CNN.

corresponds to the nature of probing strategy that leverages k-center (diversifying classes) and DeepFool (approaching decision boundary). BDPL demonstrates stronger capability against such attack. The drop of extraction rate is smaller in *CIFAR-10 w/ CNN* given that it has great complexity (over 3000 input dimensions) and low risk in current extraction attack.

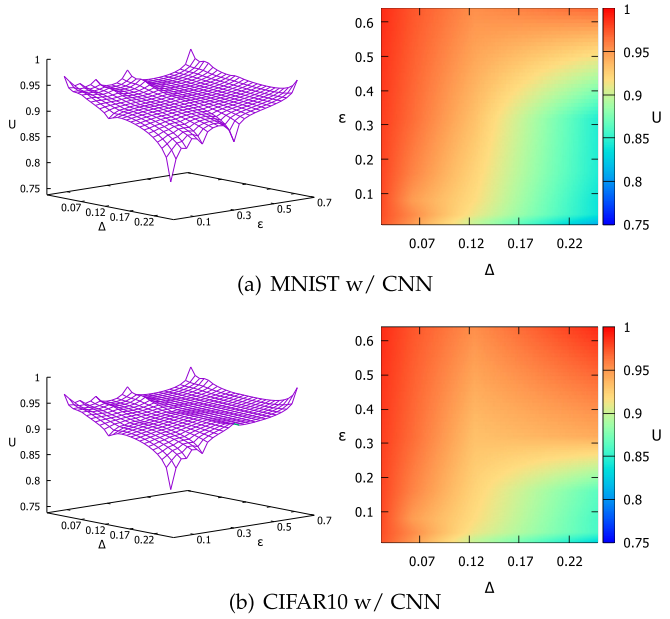
As for the utility trend, both models are saturated at over 90 percent. The perturbation stays obviously light which inhibits further drop of extraction rate. We conjecture that current defense is not entirely on its optimal performance due to high-dimensionality and great model complexity. Section 6 will further identify the limitations of BDPL and areas of improvement.

5.6.2 *Impact of ϵ and Δ on Complex Model*

In this subsection, we evaluate BDPL performance with respect to zone parameter Δ and privacy budget ϵ. In Fig. 14, we fix ϵ and vary Δ from $1/32$ to $1/4$ in BDPL. In Fig. 15, we fix Δ and vary ϵ from 0.01 to 0.64 in BDPL. We mainly plot the results on *MNIST w/ CNN* under non-probing and probing attacks.

The extraction rate is reduced more significantly on ATHS when Δ increases from $1/32$ to $1/4$. As coincided with the design of BDPL, it has bigger impact on probing strategy. Moreover, the drop of extraction rate is obviously greater on $\Delta = 1/4$ than the other 3 parameters, which indicates that a large Δ is required for effective defense on the current model. As for ϵ, when decreasing to 0.01, the change is less significant compared to that in Δ. This indicates that complex model is less sensitive against the change of perturbation (privacy budget). Overall, the extraction rates are consistently bounded and saturate as the query size increases. ϵ-BDPL still imposes a theoretical upper bound on the extraction rate and prevent full extraction. Furthermore, BDPL displays flexible control over the extraction in probing strategy.

We also evaluate the utility distribution on the same varying settings on both models. The results are reported in Fig. 16 after $20K$. The utility is saturated at over 80 percent



(a) MNIST w/ CNN

(b) CIFAR10 w/ CNN

Fig. 16. Utility versus Δ and ϵ on complex models.

under both models, which shows that BDPL has not severely degraded service quality given the significant growth of model complexity. Consistently, the peak utility is achieved when $\Delta = 1/32$ (the smallest zone size) and $\epsilon = 0.64$ (the least probability of perturbation).

To conclude, BDPL alleviates the threat of extraction rate in spite of the significant growth of victim complexity and strong attack. The change of Δ takes dominant control over the extraction rate. We will discuss the limitations and improvements in the following section.

6 Discussion

In this section, we review the evaluation of BDPL and identify areas of future improvement.

First, for attacks that only support probability-level extraction, our BDPL, which is at label-level, cannot protect against it. On the other hand, we'd argue that BDPL can still protect against attacks extraction using natural data. First, normal and natural data can also be close to decision boundary, although the ratio of sensitive queries in these attacks is lower than that in the fine-tuned ones. Second, the optimal strategy in recent studies [7], [8] leverages adversarial techniques which implicitly perform fine-tuned probing on the decision boundary. This means our BDPL can effectively protect against them, as indicated in our new experimental results against [8] in Section 5.6. Nonetheless, extraction using natural data is limited on specific model types, such as images and text, where same domain data can be easily obtained and used as query set. If the victim is a genetic model, it would be difficult to perform such extraction since genetic data is usually proprietary.

Second, we identify two core components that can be further improved for better performance in complex model. One is distance metrics and the other is perturbation mechanism.

Distance Metrics. In the current version, we adopt flipping-corner-point technique on two essential assumptions: modular design and generality. The first one allows plug-in feature for practical deployment, where service providers can tap in BDPL with the same API for users. The second one ensures compatibility with both parametric (e.g., neural network) and non-parametric models (e.g., decision tree). Unlike the classical models, deep neural networks such as image models may have high model complexity and thousands of input dimensions. Our intuition for flipping-corner technique, which comes from the L-norm ball, may have unexpected behavior in such space [20].

To improve the scalability, we can start by relaxing the first assumption and access the internals of provider's model. As such, a straightforward remedy becomes feasible by performing flipping-corner detection in the middle part of model, such as a bottleneck layer [21]. The dimensionality is greatly reduced after the intermediate representation compared to the raw input. Apart from the high-dimensionality, the manifold assumption discussed in adversarial robustness of complex model may also render flipping-corner-point technique unstable. Corner points may not flip properly when samples fall out of the manifold. As a result, L-norm distance metrics may degrade the accuracy of detection process. By relaxing the second assumption, we can leverage the gradient from parametric model to propose gradient-based distance metrics, which is more suitable under the manifold assumption. We believe it is a viable and practical solution for future improvement as complex parametric models are prevailing in machine learning services.

Perturbation Mechanism. As motivated by binary defense, randomized response is adopted as the basic framework for multiclass defense. We perform a one-vs-one approximation for each class and treat multiclass defense as a combination of binary defense. This assumption may incur imbalanced noises since only one counter class is considered. Perturbation may be concentrated on specific pairs of classes. To resolve this, a natural framework capable of categorical-value perturbation, such as k-ary randomized response [22], can be adopted for multiclass defense. Furthermore, if we extend the mechanism to be numeric-value suitable, probability-level defense becomes feasible. Nonetheless, these existing mechanisms will still need significant adaptation to satisfy ϵ-BDP before applying it to our defense. We plan to implement them in future work.

7 Related Works

There are three streams of related works, namely, machine learning model extraction, defense, and differential privacy.

Model Extraction. Machine-learning-as-a-service (MLaaS) has furnished model extraction attacks through the rich information available from prediction API. Tramer *et al.* [1] proposed extraction methods that leveraged the confidence information in the API and managed to extract the full set of model parameters using equation-solving. Papernot [3] *et al.* introduced a Jacobian-based data augmentation technique to extend queries and to train a substitute DNN. Similarly, Juuti *et al.* [7] leveraged both optimal hyperparameters and Jacobian-based data augmentation to extract models under their generalized framework. Orekondy *et al.* [9] proposed a knockoff model to steal the functionality of an image classifier and developed reinforce-based query strategy using multi-armed bandits problem. Pal *et al.* [8] further

improved the extraction attack on image model without annotated data. Their hybrid strategy combines greedy clustering with adversarial samples. Yu *et al.* [16] proposed new adversarial samples generation technqiue named FeatureFool using feature-based optimization algorithm and performed model extraction on MLaaS platform. Besides extracting model parameters, Wang *et al.* [23] also extracted the hyperparamters of a fully trained model by utilizing the zero gradient technique. Oh *et al.* [24] developed a model-of-model to infer internal information of a neural network such as layer type and kernel sizes.

Model extraction without confidence is similar to *learning with membership query* [25], [26], which learns a concept through querying membership on an oracle. This technique has been exploited by Lowd *et al.* to extract binary classifiers [5]. They used line search to produce optimized queries for linear model extraction. This technique was extended by Tramer *et al.* [1] to non-linear models such as a polynomial kernel support vector machine. They adopted adaptive techniques such as active learning to synthesize fine-tuned queries and to approximate the decision boundary of a model. Pal *et al.* [8] further improved the extraction attack using only top-1 label on image model.

Model Extraction Defense. Confidence rounding and ensemble model were shown effective against equation-solving extractions in [1]. Lee *et al.* [27] proposed perturbations using the mechanism of reverse sigmoid to inject deceptive noises to output confidence, which preserved the validity of top and bottom rank labels. Kesarwani *et al.* [6] monitored user-server streams to evaluate the threat level of model extraction with two strategies based on entropy and compact model summaries. The former derived information gain with a decision tree while the latter measured feature coverage of the input space partitioned by source model, both of which were highly correlated to extraction level. Juuti *et al.* [7] adopted a different approach to monitor consecutive queries based on the uniqueness of extraction behavior. A warning would be generated when queries deviated from a benign distribution due to malicious probing. Quiring *et al.* [28] adopted the notion of closeness-to-the-boundary in digital watermarking and applied it to protect against extraction attacks on decision trees. The defense strategy was devised from protection of watermark detector and it monitored the number of queries that fell into security margin.

Differential Privacy. Differential privacy (DP) was first proposed by Dwork [10] to guarantee the privacy of a centralized dataset with standardized mathematical notation. Duchi *et al.* [29] extended this notation to local differential privacy (LDP) for distributed data sources. Randomized response proposed by Warner *et al.* [11] is the baseline perturbation algorithm for LDP, which protects binary answers of individuals. Although differential privacy has not been used in model extraction and defense, it has been applied in several adversarial machine learning tasks. For example, Abadi *et al.* [30] introduced differentially private stochastic gradient descent to deep learning, which can preserve private information of the training set. Lee *et al.* [31] further improved its effectiveness using an adaptive privacy budget. Their approaches are shown effective against model inversion attack [2] or membership inference attack [32].

8 Conclusion and Future Work

In this paper, we propose boundary differentially private layer to defend machine learning models against extraction attacks by obfuscating the query responses. This layer guarantees boundary differential privacy in a user-specified boundary-sensitive zone. To identify sensitive queries that fall in a zone, we develop an efficient approach that uses corner points as indicators. We design boundary randomized response as the building block for perturbation algorithm, followed by a generalization to multiclass model and an adaptive version that can protect a soft margin of decision boundary. We prove such perturbation algorithm satisfies ϵ-BDP. Through extensive experimental results, we demonstrate the effectiveness and flexibility of our defense layer on protecting decision boundary while retaining high utility of the machine learning service.

For future work, we plan to propose defense more suitable for complex model and consider strong adversary with evasion. We also plan to extend our defense layer to protect against other machine learning attacks such as model evasion and inversion.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No: U1636205 and 61572413), the Research Grants Council, Hong Kong SAR, China (Grant No: 15238116, 15222118, 15218919, and C1008-16G), and a Research Project from Huawei.

References

[1] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Conf. Secur. Symp.*, 2016, pp. 601–618.

[2] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.

[3] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 506–519.

[4] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Proc. Annu. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 16–25.

[5] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 641–647.

[6] M. Kesarwani, B. Mukhoty, V. Arya, and S. Mehta, "Model extraction warning in MLaaS paradigm," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2018, pp. 371–380.

[7] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "PRADA: Protecting against DNN model stealing attacks," *IEEE Eur. Symp. Secur. Privacy*, pp. 512–527, 2019.

[8] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. K. Shevade, and V. Ganapathy, "ActiveThief: Model extraction using active learning and unannotated public data," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 865–872.

[9] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff Nets: Stealing functionality of black-box models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4949–4958.

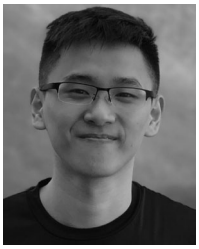
[10] C. Dwork, "Differential privacy," in *Proc. 33rd Int. Colloq. Automata Languages Program.*, 2006, pp. 1–12.

[11] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *J. Amer. Statist. Assoc.*, vol. 60, no. 309, pp. 63–66, 1965.

[12] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[13] T. W. Smith, P. Marsden, M. Hout, and J. Kim, "General social survey," 2013. [Online]. Available: https://gss.norc.org/Get-The-Data

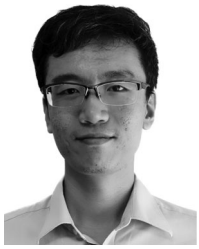
[14] D. Harris and S. Harris, *Digital Design and Computer Architecture*. San Mateo, CA: Morgan Kaufmann, 2007.
[15] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/
[16] H. Yu, K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin, "Cloudleak: Large-scale deep learning models stealing through adversarial examples," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020.
[17] Y. LeCun, "The MNIST database of handwritten digits," Accessed: Apr. 5, 2020. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[18] A. Krizhevsky, "The CIFAR-10 dataset," Accessed: Apr. 5, 2020. [Online]. Available: https://www.cs.toronto.edu/ kriz/cifar.html
[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[20] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional spaces," in *Proc. Int. Conf. Database Theory*, 2001, pp. 420–434.
[21] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.
[22] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *Proc. 33nd Int. Conf. Mach. Learn.*, 2016, pp. 2436–2444.
[23] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 36–52.
[24] S. J. Oh, M. Augustin, B. Schiele, and M. Fritz, "Towards reverse-engineering black-box neural networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[25] D. Angluin, "Queries and concept learning," *Mach. Learn.*, vol. 2, pp. 319–342, 1987.
[26] L. G. Valiant, "A theory of the learnable," in *Proc. ACM Symp. Theory Comput.*, 1984, pp. 436–445.
[27] T. Lee, B. Edwards, I. Molloy, and D. Su, "Defending against model stealing attacks using deceptive perturbations," 2018, *arXiv:1806.00054*.
[28] E. Quiring, D. Arp, and K. Rieck, "Forgotten siblings: Unifying attacks on machine learning and digital watermarking," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2018, pp. 488–502.
[29] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proc. IEEE Symp. Foundations Comput. Sci.*, 2013, pp. 429–438.
[30] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
[31] J. Lee and D. Kifer, "Concentrated differentially private gradient descent with adaptive per-iteration privacy budget," in *Proc. ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2018, pp. 1656–1665.
[32] R. Shokri, M. Stronati, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.
[33] Q. Ye, H. Hu, N. Li, X. Meng, H. Zheng, and H. Yan, "Beyond value perturbation: Differential privacy in the temporal setting," in *Proc. IEEE Int. Conf. Comput. Commun.*, to be published.
[34] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Towards locally differentially private generic graph metric estimation," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 1922–1925.



Huadi Zheng (Student Member, IEEE) receives the BEng degree from the School of Data and Computer Science, Sun Yat-sen University, China, in 2012. Currently he is working toward the PhD degree with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. His research interests include mobile side-channel security, data privacy, and adversarial machine learning.

Qingqing Ye (Member, IEEE) received the PhD degree in computer science from the Renmin University of China, in 2020. She is a research assistant professor with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University. She has received several prestigious awards, including China National Scholarship, Outstanding Doctoral Dissertation Award, and IEEE S&P Student Travel Award. Her research interests include data privacy and security, and adversarial machine learning.

Haibo Hu (Senior Member, IEEE) is an associate professor with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University and the programme leader of BSc (Hons) in Information Security. His research interests include cybersecurity, data privacy, Internet of Things, and adversarial machine learning. He has published more than 80 research papers in refereed journals, international conferences, and book chapters. As principal investigator, he has received more than 20 million HK dollars of external research grants from Hong Kong and mainland China as of year 2020. He has served in the organizing committee of many international conferences, such as ACM GIS 2021, 2020, IEEE ICDSC 2020, IEEE MDM 2019, DASFAA 2011, DaMEN 2011, 2013, and CloudDB 2011, and in the programme committee of dozens of international conferences and symposiums. He is the recipient of a number of titles and awards, including IEEE MDM 2019 Best Paper Award, WAIM Distinguished Young Lecturer, ICDE 2020 Outstanding Reviewer, VLDB 2018 Distinguished Reviewer, ACM-HK Best PhD Paper, Microsoft Imagine Cup, and GS1 Internet of Things Award.



Chengfang Fang received the PhD degree from the National University of Singapore before joining Huawei. He has been working on security and privacy protection in several areas including machine learning, Internet of Things, mobile device, and biometrics for more than 10 years. He has published more than 20 research papers and obtained 15 patents in the domain. He is currently a principle researcher at Huawei Singapore Research Center.

Jie Shi received the PhD degree from the Huazhong University of Science and Technology, China. He is a principal researcher at Huawei Singapore Research Center. His research interests include trustworthy AI, machine learning security, data security and privacy, IoT security, and applied cryptography. He has more than 10 years' research experience and has published more than 30 research papers in refereed journals and international conferences.