Prediction Poisoning: Utility-Constrained Defenses Against Model Stealing Attacks

Tribhuvanesh Orekondy\textsuperscript{1} Bernt Schiele\textsuperscript{1} Mario Fritz\textsuperscript{2}
\textsuperscript{1} Max Planck Institute for Informatics \textsuperscript{2} CISPA Helmholtz Center for Information Security Saarland Informatics Campus, Germany

Abstract

With the advances of ML models in recent years, we are seeing an increasing number of real-world commercial applications and services e.g., autonomous vehicles, medical equipment, web APIs emerge. Recent advances in model functionality stealing attacks via black-box access (i.e., inputs in, predictions out) threaten the business model of such ML applications, which require a lot of time, money, and effort to develop. In this paper, we address the issue by studying defenses for model stealing attacks, largely motivated by a lack of effective defenses in literature. We work towards the first defense which introduces targeted perturbations to the model predictions under a utility constraint. Our approach introduces perturbations targeted towards manipulating the training procedure of the attacker. We evaluate our approach on multiple datasets and attack scenarios across a range of utility constrains. Our results show that it is indeed possible to trade-off utility (e.g., deviation from original prediction, test accuracy) to significantly reduce effectiveness of model stealing attacks.

1 Introduction

Machine learning (ML) models are increasingly being deployed in a variety of real-world applications e.g., home assistants, autonomous vehicles, commercial cloud APIs. The ML models in such applications are valuable intellectual property of their creators, as developing them for commercial use is a product of intense labour and monetary effort. Hence, it is vital to preemptively identify and control threats from an adversarial lens focused at such models. To this end, the community recently has taken great effort to address such issues by studying various subproblems such as evasion attacks \cite{7,15,17,26}, membership inference \cite{22,24}, model stealing \cite{19,27,29}, and data poisoning \cite{2,13,23}. In this work, we focus on model stealing attacks, which involves an attacker attempting to reconstruct a black-box victim ML model.

While knowledge distillation (KD) \cite{3,8} has led to advances in many problems (e.g., object detection \cite{4}, classification \cite{51}), variants of it are remarkably effective to perform model stealing. Here, the attacker (student model) queries inputs to the victim blackbox (teacher model) to obtain respective posterior predictions (soft targets). The stolen model is then trained on the ‘transfer set’ of inputs and their corresponding posterior predictions. Recently, it was shown \cite{20} that such attacks are surprisingly effective, even when the inputs are from an independent distribution (e.g., ImageNet) and different from that of the blackbox (e.g., birds classifier).

In this paper, we are motivated to study defenses against model attacks as existing defenses have been largely ineffective. Specifically, the papers that propose stealing \cite{20,27} find the attacks resilient to various forms of information truncation schemes (e.g., rounding posteriors, returning top-\textit{k} predictions). In the most extreme case, we too show that attacks are highly effective, even when the attacker is provided the minimum information for the input query: a single most-confident label. Hence, literature suggests protection against model stealing is currently not feasible, making model
stealing attacks a threat to commercial ML applications, most of which allow a form of black-box query access.

How do we work towards effective defenses for model functionality stealing attacks? We study defenses in a black-box setup, where the only means of interaction between the attacker and the defender are via black-box access: attacker inputs a query, defender returns posterior predictions. Defending in such a scenario is especially challenging since the utility of the prediction is intertwined with information of model functionality. We work towards defenses in a utility-constrained framework, wherein the defender trades away some utility of the prediction to adversely impact the attacker. Towards this goal, we list our contributions as:

- We propose the first utility-constrained defense framework against model stealing attacks.
- We present an approach which introduces bounded perturbations to the predictions, by introducing targeted noise to degrade first-order approximations while training stolen models.
- Our extensive experiments on multiple datasets show that it is, under certain circumstances and in principle, possible to defend against such attacks and thereby significantly reduce the performance achieved by the stolen model.

2 Related Work

In this section, we review related literature on model stealing attacks and defenses.

Model Stealing Attacks. Stealing various aspects of a black-box model have been proposed: parameters [27], hyperparameters [29], architecture [19], data membership information [24]. In this paper, we focus on functionality i.e., the attacker intends to maximize test performance of the stolen model. Two closely-related flavours of model stealing attacks exist: distillation-based [8, 20, 27] and data-augmentation-based [11, 21]. In case of the latter, stealing the model is an intermediate step towards the final goal of crafting adversarial examples. In this paper, we primarily use the distillation-based approaches whose final goal is model stealing; they have shown to be highly effective [27], even for stealing state-of-the-art vision models [20] despite minimal assumptions.

Identifying Stolen Models via Watermarks. Watermarking machine learning models [1, 9, 28, 30] involves the owner training the model with an additional function e.g., uniquely activating to a set of secret watermarked inputs. However, the primary purpose of watermarking is to prove post-hoc ownership verification, rather than defend against stealing attacks. In contrast, we seek a preemptive defense which adversely affects the performance of the stolen model.

Detecting Model Stealing Attacks. One proposed way to prevent model stealing attacks is by characterizing consecutive input queries and detecting malicious intent. In [11], the authors identify query inputs of augmentation-based model stealing attacks [11, 21] as having low inter-query input distances in euclidean space (since the augmented inputs lie in a small $L_p$-bounded $\epsilon$-ball). However, this has a drawback that the attacker can query using multiple accounts, or simply increase the size of the $\epsilon$-ball to avoid detection. In contrast, we seek to poison the output predictions; a task that is parallel (also which can be combined) with detection strategies.

Train-time Poisoning Attacks. In data poisoning attacks [2, 13, 14, 18, 23], an attacker intends to corrupt a minimal set on input training examples that the victim eventually uses to train a model. Predominantly, the corruption is done by perturbing inputs and optionally flipping the label. This is reminiscent of our problem, but now with the defender intending to poison the training data of the attacker. However, unlike data poisoning attacks, we do not have the ability to perturb inputs nor assume prior access to the pool of examples that the attacker intends to query.

Altering Predictions. Existing literature that study defenses [16, 20, 27] show only limited success. The strategies proposed in these simple defenses reduce or perturb the posteriors, while maintaining the rank of the top-1 ‘argmax’ label i.e., accuracy of the task. However, consistent with their observations, we too find this strategy has limited effect on a model stealing attacker. Hence, in this paper, we work towards a utility-constrained defense allowing the defender to trade-off utility (e.g., perturbation magnitude, accuracy) for privacy i.e., poor performance of the adversary.
3 Problem Statement

In this section, we first recap model stealing and proceed towards formalizing the defense objective.

Model Functionality Stealing. Model stealing attacks [20, 27] are cast as an interaction between two parties: a victim/defender V (teacher model) and an attacker A (student model). The only means of communication between the parties are via black-box queries: attacker queries inputs \( x \in X \) (typically images) and defender returns a posterior probability distribution \( y \in \Delta^K = P(y|x) = F_V(x) \), where \( \Delta^K = \{y \geq 0, \sum y = 1\} \) is the probability simplex over \( K \) classes (we use \( K \) instead of \( K-1 \) for notational convenience). Using black-box access to the defender’s model, the attacker trains a stolen model \( F_A \) using the input-predictions pairs, i.e., the ‘transfer set’ of queries \( D_{\text{transfer}} = \{(x_i, y_i)\}_{i=1}^B \), with the end-goal of maximizing accuracy of \( F_A \) on the defender’s test set \( D_{\text{test}} \), which we denote as \( \text{Acc}(F_A, D_{\text{test}}) \).

Data-limited Attacker. In model stealing, in contrast to knowledge distillation (KD), students (attackers) are justifiably data-limited i.e., do not have access to the entirety of labeled data as that of the teacher (defender). What then makes for the inputs \( \{x \sim P_A(X)\} \) of the attacker? There are two flavours: (a) independent distribution: [20, 27] samples inputs from some distribution (e.g., ImageNet for images, uniform noise) independent to input data used to train the victim model; and (b) synthetic set: [21] augment a limited set of input data by adding small input perturbations, that makes the stolen model ideal as a surrogate for crafting adversarial examples. However, since the synthetic set involves perturbations of images within a small \( \epsilon \)-ball, they are potentially easier to detect and therefore defend [11]. We choose (a), which is specifically targeted towards model stealing, also which results in strongest attack performance and scales to complex state-of-the-art models.

Defender’s Objective. We now focus on the defender (the victim blackbox), who, given an input \( x \), intends to return posterior predictions \( y = F_V(x) \). As part of the defense, the defender now returns perturbed predictions \( \hat{y} = F_V^\delta(x) = y + \delta \) s.t. \( y, y \in \Delta^K \). The defender has two objectives: (i) Privacy objective: to reduce the final stolen model accuracy of the attacker \( \text{Acc}(F_A, D_{\text{test}}) \), (ii) Utility objectives: we explore two choices which measure the distance between posteriors \( \text{dist}(y, \hat{y}) \in \mathbb{R}^+ \) as: (a) dist-lp \( y, \hat{y} = ||y - \hat{y}||_p = \epsilon \), also which serves as a proxy to victim’s accuracy; and optionally (b) dist-argmax \( y, \hat{y} = 0 \) if \( \arg \max_{y} y_k = \arg \max_{\hat{y}} \hat{y}_k \) and, \( \infty \) otherwise. In our experiments, we evaluate privacy as \( \text{Acc}(F_A, D_{\text{test}}) \). For utility, we consider both accuracy \( \text{Acc}(F_V^\epsilon, D_{\text{test}}) \) and perturbation magnitude \( \epsilon \). Later in the paper (Section 5.2), we also evaluate attacker’s strategies at circumventing the defender’s objectives.

Model Stealing Defense as an Optimization Problem As a defender, we would ideally like to solve a bilevel optimization problem to poison [2, 13, 14, 18] the training data of the attacker:

\[
\max_{\hat{y}, w_A} - \text{Acc}(F_A(\cdot; w_A)) \quad \text{(Privacy objective)} \tag{1}
\]

\[
s.t. \quad \text{dist}(y_i, \hat{y}_i) \leq \epsilon \quad \text{(Utility objective)} \tag{2}
\]

where \( w_A = \arg \min_{w} L(F_A(\cdot; w), D_{\text{transfer}}) \tag{3} \)

\[
D_{\text{transfer}} = \{(x_i, \hat{y}_i)\}_{i=1}^B
\]

The idea here is to poison the attacker’s training data of the stolen model \( F_A \) by introducing targeted perturbations to predictions. However, this formulation faces few challenges: (a) the defender does not have prior access to the pool of inputs \( \{x_i\} \) the attacker wishes to query; and (b) the accuracy objective (Eq. 1) is a complex function (Eq. 3) of the predictions returned by the defender.

To tackle these challenges, in this paper, given some input query \( x \) and prediction \( y = F_V(x) \), we wish to solve a simpler problem:

\[
\max_{\hat{y} \in \Delta^K} - \text{ProxyAcc}(F_A(\cdot; w_A), \hat{y}) \quad \text{(Privacy objective)} \tag{4}
\]

\[
s.t. \quad \text{dist}(y, \hat{y}) \leq \epsilon \quad \text{(Utility objective)} \tag{5}
\]

where we seek to poison a proxy to attacker’s training procedure of the stolen model by introducing targeted perturbations \( \hat{y} \). In the next section, we elaborate on our choice of \( \text{ProxyAcc}(\cdot) \) and discuss an approach to optimize this problem.
4 Approach: Maximizing Angular Deviation between Gradients

In this section, we present our training poisoning objective (Eq. 4) and work towards an approach to efficiently solve the utility-constrained model stealing defense problem (Eq. 4-5).

Motivation: Targeting First-order Approximations. We identify that the attacker eventually optimizes parameters of a stolen (student) model \( F(\cdot; w) \) (we drop the subscript \( \cdot; A \) for readability) to minimize the loss on training examples \( \{ (x_i, y_i) \} \). Common to a majority of optimization algorithms is estimating the first-order approximation of the empirical loss, by computing the gradient of the loss w.r.t. the model parameters \( w \in \mathbb{R}^D \):

\[
   u = -\nabla_w L(F(x; w), y)
\]

Maximizing Angular Deviation (MAD). The core idea of our approach is to perturb the posterior probabilities \( y \) which results in an adversarial gradient signal that maximally deviates from the original gradient (Eq. 6). More formally, we add targeted noise to the posteriors which results in a gradient direction:

\[
   a = -\nabla_w L(F(x; w), \hat{y})
\]

to maximize the angular deviation between the original and the poisoned gradient signals:

\[
   \max_a 2(1 - \cos \angle(a, u)) = \max_a ||\hat{a} - \hat{u}||_2^2 \quad (\hat{a} = a/||a||_2, \hat{u} = u/||u||_2)
\]

Given that the attacker model is trained to match the posterior predictions, such as by minimizing the cross-entropy loss \( L(y, \hat{y}) = -\sum_k \hat{y}_k \log y_k \) we can rewrite Equation (7) as:

\[
   a = -\nabla_w L(F(x; w), \hat{y}) = \nabla_w \sum_k \hat{y}_k \log F(x; w)_k = \sum_k \hat{y}_k \nabla_w \log F(x; w)_k = G^T \hat{y}
\]

where \( G \in \mathbb{R}^{K \times D} \) represents the Jacobian over log-likelihood predictions \( F(x; w) \) over \( K \) classes w.r.t. parameters \( w \in \mathbb{R}^D \). By similarly rewriting Equation (6), substituting them in Equation (8) and including the constraints, we arrive at our poisoning objective:

\[
   \max_{\hat{y}} \left( \frac{G^T \hat{y}}{||G^T \hat{y}||_2} - \frac{G^T y}{||G^T y||_2} \right)^2 \quad (= H(\hat{y}))
\]

\[
   \text{s.t.} \quad \hat{y} \in \Delta^K \quad (\text{Simplex constraint})
\]

\[
   \text{dist}(y, \hat{y}) \leq \epsilon \quad (\text{Utility constraint})
\]

Solving a Convex Relaxation. Directly optimizing this objective presents a challenge of constrained maximization of a non-convex objective (orange line in Fig. 1). Hence, we instead seek to solve a convex relaxation, \( \hat{H}(\hat{y}) = ||G^T \hat{y} - G^T y||_2^2 \) (blue line in Fig. 1).

Temporarily ignoring the utility constraint (Equation 12), the global maximum \( y^* \) for the relaxed objective is one of the extremes of the (convex) simplex feasible set \( \Delta^K \) (see [10], Theorem 1), which we denote as ext(\( \Delta^K \)). In our scenario, \( y^* \) can be tractably computed, since ext(\( \Delta^K \)) has only \( K \) elements. We denote the \( k \)-th extreme as \( y_k \in \Delta^K \) where \( y_k = 1 \).

Enforcing dist-lp(·) objective: MAD. We estimate the perturbed posteriors \( \hat{y} \) as a linear combination of the posteriors and the global maximum: \( \hat{y} = (1 - \alpha)y + \alpha y^* \), where \( \alpha \) is selected such that the utility constraint (Equation 12) is satisfied. This can be done by solving for \( \alpha \) using bisection method, or solving for \( \text{dist}(y, \hat{y}) = \epsilon \). An analytic solution exists for the latter in case of \( L_p \) distances: \( \alpha^* := \max \left\{ \frac{\epsilon}{||y - y^*||_p}, 1 \right\} \).

Including dist-argmax(·) objective: MAD-argmax. We now introduce a variant to additionally encode the accuracy-preserving utility objective \( \text{dist-argmax}(y, \hat{y}) \) of the defender. We do this by
preserving the most confident ‘argmax’ label: \( k = \operatorname{arg\,max}_k y_k = \operatorname{arg\,max}_k \tilde{y}_k \) in the perturbed prediction. Within our approach, this amounts to restricting the search space in the simplex \( \Delta^K \) (Equation 11). We absorb the \( k \)-th label-preserving constraint by iterating over the extremes of the probability simplex \( \Delta^K = \{ y \geq 0, 1^T y = 1, y_k \geq y_j, k \neq j \} \subseteq \Delta^K \).

5 Experimental Results

We now introduce our experimental setup in Section 5.1, followed by results in Section 5.2.

5.1 Experimental Setup

Victim Models and Datasets. We set up three victim models (see Table 1), each model trained on a popular image classification dataset. We train each model on their respective train sets (60k examples for MNIST/FashionMNIST, 50k for CIFAR10) and evaluate on their test sets (10k examples in each dataset). For MNIST and FashionMNIST, we use a LeNet-like architecture with two convolutional layers (5 × 5 kernels; stride 1; 20, 50 channels), each followed by max pooling (2 × 2 kernel), and two fully connected layers; for CIFAR10, we use VGG-16 \(^{25}\). We train all models from scratch for 50 epochs (100 for CIFAR10) using SGD with a learning rate of 0.01 and 0.5 momentum.

Attack Model. We use a distillation-based attack strategy \(^{20}\) (an extension to \(^{27}\)), which in our preliminary experiments yielded the strongest attack results. The attack is split into two steps: (a) obtaining a transfer set \( D^{\text{transfer}} = \{ (x_i, y_i) \}_{i=1}^B \) by querying images \( x_i \sim P_A(X) \) to the victim model; and (b) training a stolen model \( F_\tilde{A} \) to minimize the cross-entropy loss on \( D^{\text{transfer}} \). Our choices of \( P_A(X) \) for the data-limited adversary for each of the victim models is listed in Table 1. EMNIST \(^5\) is a dataset similar to MNIST containing handwritten characters with upper-/lower-case alphabets and digits. EMNISTLetters is a subset of EMNIST limited to alphabets. By default, we use the victim architecture for stolen models, but with different initializations. This minimally impacts attack and defense performances (experiments for different choices of architectures in Section 5.2). Due to the complex parameterization of VGG-16 (14M+), we initialize the weights from a pretrained TinyImageNet model (except for the last FC layer, which is trained from scratch). We train all stolen models in (b) for 50 epochs using SGD with a learning rate of 0.1 and 0.5 momentum.

Effectiveness of Attacks. In Figure 2 we present the test performance of the undefended victim \( V \) model (dotted horizontal lines) and the attacker ‘A’s stolen model (solid lines) for multiple budgets (equal to number of queries to the victim model and size of transfer set). We observe for all three victim models (represented in three colours), using just 50K black-box queries, the attacker is highly effective, recovering up to 99.6% accuracy of the undefended model.

How Good are Existing Defenses? Most existing defenses in literature \(^{16, 20, 27}\) perform some form of information reduction on the posterior probabilities e.g., rounding, returning top-\( k \)-labels; all these defenses preserve the rank of the most confident label. We now evaluate our model stealing attack on the extreme end of information truncation, wherein the defender returns just the top-1 ‘argmax’ label. This strategy illustrates a rough lower bound on the strength of the attacker when using existing defenses. In Figure 2 we observe that the attacker is minimally impacted, with a 3.3% mean accuracy difference between undefended (solid lines) and argmax-defended (dash-dotted lines) models. We find defenses in existing literature achieve limited success against model stealing attacks.

Defenses: Setup and Evaluation. We use a randomly initialized proxy model with the same architecture as \( F_V \) (defender’s model) to estimate the jacobian \( G \) (Eq. 10). Compared with other initialization strategies (e.g., partially or completely trained), we found random initialization provide a better gradient signal to poison the posteriors. Furthermore, we compute the jacobian \( G \) over
parameters of the final fully connected layer of the network. In preliminary experiments, this yielded better results over using all parameters of the network.

We evaluate all defenses on a privacy-vs-utility curve at various operating points of the defense. The privacy metric is fixed to accuracy of the stolen model on the held-out test data. We use two utility metrics: (a) perturbation magnitude $\epsilon$: measured as $L_1$ distance $||y - \hat{y}||_1$; and (b) accuracy: test-accuracy of the defended model producing perturbed predictions.

We compare our approaches against two methods: reverse-sigmoid \cite{16} and random-noise. Reverse-sigmoid softens the posterior distribution and introduces ambiguity among non-argmax probabilities. For this method, we evaluate privacy and utility metrics for the defense operating at various choices of their hyperparameter $\beta \in [0, 1]$, while keeping their dataset-specific hyperparameter $\gamma$ fixed (MNIST: 0.2, FashionMNIST: 0.4, CIFAR10: 0.1). For controlled random-noise, we add uniform random noise $\delta_z$ on the logit prediction scores ($\tilde{z} = z + \delta_z$, where $z = \log(\frac{y}{1 - y})$), enforce utility by projecting $\delta_z$ to an $\epsilon_z$-ball \cite{6}, and renormalizing probabilities $\hat{y} = \frac{1}{1 + e^{-\epsilon}}$.

5.2 Results

We now discuss results of the proposed defenses.

**How effective are the defenses?** We evaluate approaches MAD and MAD-argmax on a privacy ($x$-axis) vs. utility ($y$-axis) curves, as shown in Figure \ref{fig:utility}. Keeping the privacy objective (test accuracy of attacker) fixed, we display individual curves for two utility metrics – perturbation $\epsilon$ (Fig. \ref{fig:utility}a) and test-accuracy of defender (Fig. \ref{fig:utility}b). Each point denotes the resulting performance of the model stealing attack with varying utility budgets ($\epsilon$ in case of our approaches). From Figure \ref{fig:utility}, we make the following observations.

(i) **Utility objective = $L_1$ distance** (Fig. \ref{fig:utility}a): Although random-noise and reverse-sigmoid reduce attacker’s accuracy, it involves larger perturbations. In contrast, MAD and MAD-argmax provides similar privacy with significantly lesser perturbation, especially at lower magnitudes of perturbation. For instance, on MNIST (first column), MAD ($L_1 = 0.95$) reduces the accuracy of the attacker to under 80% with 0.63× the perturbation as that of reverse-sigmoid and random-noise ($L_1 \approx 1.5$).

(ii) **Utility objective = argmax-preserving** (Fig. \ref{fig:utility}b): By setting a hard constraint on retaining the label of the predictions, we find MAD-argmax and reverse-sigmoid successfully reduce the performance of the attacker by at least 20% across all datasets. Remarkably, we find MAD-argmax in addition achieve this objective by introducing lesser distortion to the predictions compared to reverse-sigmoid. For instance, in Fig. \ref{fig:utility}b, we find MAD-argmax consistently reduce the attacker accuracy to the same
amount at lesser $L_1$ distances. In reverse-sigmoid, we attribute the large $L_1$ perturbations to a shift in posteriors towards a uniform distribution e.g., mean entropy of perturbed predictions is 3.02 ± 0.16 (max-entropy = 3.32) at $L_1=1.0$ for MNIST; in contrast, MAD-argmax displays a mean entropy of 1.79 ± 0.11. However, common to label-preserving strategies is a pitfall that the most-confident label is retained. How does the attacker perform if he/she were to train a stolen model discarding the posteriors and only using the most-confident label for the query inputs? In Figure S2 (see overlapping red and yellow cross-marks), we find that the attacker is able to completely recover the original performance of the stolen model. In line with our previous experiment (Figure 2), results indicate the label-preserving objective can be circumvented by the adversary by discarding the posteriors and using the most-confident label during training. We now ask whether the defender can trade-off utility (test accuracy or $L_1$ perturbation) to significantly reduce the attacker’s performance.

(iii) A privacy-utility trade-off (Fig. 3b): Across all models, we find MAD provides reasonable operating points (above the diagonal), where defender displays higher test accuracies compared to the attacker. In Fig. 3, for instance, at $\epsilon=0.95$ on CIFAR10, we observe the defender’s test accuracy (dotted lines) drop from 92% to 88% (a 4% absolute decrease). Whereas, the attacker’s accuracy (solid lines) drops 6× from 92% to 54% (a 24% decrease). Moreover, as seen in Fig. S2, MAD is robust even when the attacker solely uses the argmax labels while training. Our results indicate promising operating points, where the defender can trade utility for a significant gain in privacy.

How much angular deviation does MAD introduce? To estimate the angular deviation between the true and the perturbed gradient, we conduct an experiment given the knowledge of the true gradient direction at each training step. We simulate this by training a white-box attacker using online SGD (LR=0.001) over $N$ iterations using $B$ distinct images to query and a batch size of 1. At each step $t$ of training, the attacker queries a randomly sampled input $x_t$ to the defender model and backpropagates the loss resulting from $\hat{y}_t$. The perturbation $\hat{y}_t$ here is crafted having exact knowledge of the attacker’s parameters. We evaluate the angular deviation between gradients with ($a$) and without ($u$) the perturbation.

In Figure 5, we visualize a histogram of deviations: $\theta = \arccos \frac{u \cdot a}{||u|| ||a||}$, where $u = \nabla_w L(w_t, \hat{y}_t)$ and $a = \nabla_w L(w_t, y_t)$). We observe: (i) although our perturbation space is severely restricted (a low-dimensional probability simplex), we can introduce surprisingly high deviations (0-115°) in the high-dimensional parameter space of the VGG16; (ii) for $\epsilon$ values at reasonable operating points which preserves the defender’s accuracy within 10% of the undefended accuracy (e.g., $\epsilon \in [0.95, 0.99]$ for CIFAR10), we see deviations with mean 24.9° (yellow bars in Fig. 5). This indicates that the perturbed gradient on an average leads to a slower decrease in loss function; (iii) on the extreme end, with $\epsilon = \epsilon_{\max} = 2$, on an average, we find the perturbations successfully flips (>90°) the gradient direction leading to an increase on the test loss, a seen in Figure 6 (blue line). We find our approach considerably influences the gradient direction.

Ablative Analysis. We present an ablation analysis of our approach in Figure 7. In this experiment, we compare our approach MAD and MAD-argmax to: (a) MAD-relax: To maximize angular deviation, inner maximization term (Eq. 9) is computed over the convex relaxation instead of the true objective; (b) $G = I$: We substitute the jacobian $G$ (Eq. 10) with a $K \times K$ identity matrix; and (c) $y^*=$rand: Inner maximization term (Eq. 9) returns a random extreme of the simplex. Note that (b) and (c) does not use the gradient information to perturb the posteriors.
We now explore various strategies an attacker can use to circumvent the defense. To this end, we evaluate the following strategies: (a) argmax: attacker uses only the most-confident label during training; (b) arch*: attacker trains other choices of architectures; (c) nquery: attacker queries each image multiple times; (d) nquery+aug: same as (c), but with random cropping and horizontal flipping; and (e) opt*: attacker uses an adaptive LR optimizer e.g., ADAM \cite{adam}. We perform experiments with these strategies by keeping the budget fixed at $B=50K$.

We present results over the subversion strategies in Figure 8. Remarkably, we find our defense robust to above strategies. Our results indicate that the best strategy for the attacker to circumvent our defense is to discard the probabilities and rely only on the most confident label to train the stolen model. For instance at $L_1=0.95$, by discarding the probabilities, the attacker can increase accuracy of the stolen model from 54\% ($0.62 \times$ defender’s test accuracy) to 69\% ($0.78 \times$).

### 6 Discussion

Our approach shows promising results: there exist trade-offs where the defender can trade utility (e.g., deviation from original prediction) to significantly reduce performance of the attacker’s stolen model. However, we acknowledge that a trade-off is not an ideal solution; some commercial ML applications might lose an edge over competition with a slight decrease in performance. We also identify that although the proposed defense significantly reduces the attacker’s test accuracy, this occurs at reasonably large deviations to posteriors ($\epsilon > 0.5$), which may not be ideal in some scenarios.

In spite of certain limitations with the trade-offs, we make some key contributions. We propose the first defense that shows encouraging results against model stealing attacks, either by significantly slowing down the attack, or even potentially fully prevent it in the future, as we show that gradient signals can be poisoned via targeted perturbed predictions.

### 7 Conclusion

In this work, we were motivated by limited success of existing defenses against models stealing attacks. We address it by studying defenses as a utility-constrained optimization problem of a privacy objective. We proposed as a proxy to the true privacy objective of minimizing attacker’s test accuracy, an approach which poisons the victim’s training procedure by perturbing the predictions. Within our framework of utility-constrained defenses for model stealing, we find our proposed approaches exhibit promising operating points for defenses against model stealing attacks.
Acknowledgement. This research was partially supported by the German Research Foundation (DFG CRC 1223). We thank Paul Swoboda and David Stutz for helpful discussions.

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 USENIX Security, 2018.

[2] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In ICML, 2012.

[3] Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In KDD, 2006.

[4] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Mannohman Chandraker. Learning efficient object detection models with knowledge distillation. In NeurIPS, 2017.

[5] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. arXiv preprint arXiv:1702.05373, 2017.

[6] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In ICML, 2008.

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

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv:1503.02531, 2015.

[9] Dorjan Hitaj and Luigi V Mancini. Have you stolen my model? evasion attacks against deep neural network watermarking techniques. arXiv preprint arXiv:1809.00615, 2018.

[10] Karla Leigh Hoffman. A method for globally minimizing concave functions over convex sets. Mathematical Programming, 20(1):22–32, 1981.

[11] Mika Juuti, Sebastian Szyller, Alexey Dmitrenko, Samuel Marchal, and N Asokan. Prada: Protecting against dnn model stealing attacks. In Euro S&P, 2019.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2014.

[13] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In ICML, 2017.

[14] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. arXiv preprint arXiv:1811.00741, 2018.

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533, 2016.

[16] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. Defending against model stealing attacks using deceptive perturbations. S&P Deep Learning and Security (DLS) Workshop, 2018.

[17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In ICLR, 2018.

[18] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In CCS AISec workshop, 2017.

[19] Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In ICLR, 2018.

[20] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In CVPR, 2019.

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

[22] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In NDSS, 2019.
[23] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In NeurIPS, 2018.

[24] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In S&P, 2017.

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

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

[27] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In USENIX Security, 2016.

[28] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin’ichi Satoh. Embedding watermarks into deep neural networks. In ICMR. ACM, 2017.

[29] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In S&P, 2018.

[30] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In Asia CCS. ACM, 2018.

[31] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning. In CVPR, 2018.
Appendices

A Contents

The appendix contains:

A. Contents (this section)
B. Detailed Algorithm
C. Extension of Existing Plots
   1. Budget vs. Accuracy
   2. Argmax-attacker
   3. Angular Deviations
   4. Ablation Experiments

B Detailed Algorithm

We present a detailed algorithm (see Algorithm 1) for the approach described in Section 4 for in the main paper.

```
1 Function PerturbedPredict-MAD(x):
   | Input: Input data x, model to defend \( F_V(\cdot; w_V) \), proxy attacker model \( F(\cdot; w) \)
   | Output: Perturbed posterior probability \( \tilde{y} \in \Delta^K \) s.t. dist(\( \tilde{y}, y \)) \leq \epsilon
2 y := F_V(x; w_V) // Obtain K-dim posteriors
3 G := \nabla_w \log F(x; w) // Pre-compute (K x D) Jacobian
4 y* := arg max_{y_k \in ext(\Delta^K)} \| G^T y_k \|_2 - \| G^T y \|_2^2 // Alternatively ext(\Delta^K)
   for MAD-argmax
5 Define \( h(\alpha) = (1 - \alpha)y + \alpha y^* \)
6 \alpha^* := arg max_{\alpha \in [0, 1], \text{dist}(\cdot) \leq \epsilon} \text{dist}(h(\alpha), y^*) // Find optimal step-size via
   bisection, or OptStep(.) for \( L_p \) norms
7 \tilde{y} := h(\alpha^*) // Perturbed probabilities
8 return \tilde{y}
9
10 Function OptStep(y, y', \epsilon, p):
11 | \alpha^* := max \left\{ \frac{\epsilon}{\| y - y' \|_p}, 1 \right\}
12 return \alpha^*
```

Algorithm 1: MAD Defense. To supplement approach in Section 4.

C Additional Plots

In this section, we extend existing results in the main paper to cover all datasets.

C.1 Budget vs. Accuracy

We plot the budget (i.e., number of distinct black-box attack queries to the defender) vs. the test accuracy of the defender/attacker in Figure S1. The figure supplements the discussion found in Section 5.2 of the main paper under “How effective are the defenses?”.
C.2 Attacker argmax

In Figure S2, we perform the privacy-vs-utility evaluation (similar to Fig. 3b in the main paper) under a special situation: the attacker discards the probabilities and only uses the top-1 ‘argmax’ label to train the stolen model. Relevant discussion can be found in Section 5.2 of the main paper under “How effective are the defenses?”.

C.3 Angular Deviations

In the white-box attacker setting, we plot the angular deviations obtained over all defender models in Figure S3. The bottom row displays test-loss of the attacker evaluated during online training. Discussions for this experiment in Section 5.2 of the main paper under “How much angular deviation does MAD introduce?”.

C.4 MAD Ablation Experiments

We present the ablation experiments covering all defender models in Figure S4. Relevant discussion is available in Section 5.2 of the main paper under “Ablative Analysis”.

Figure S1: Budget vs. Test Accuracy. Supplements Fig. 3c in the main paper.

Figure S2: Attacker argmax. Supplements Fig. 4 in the main paper.
Figure S3: Histogram of Angular Deviations. Supplements Fig. 5,6 in the main paper. The test-loss during of the attacker model for each of the histograms (over multiple $\epsilon$ values) are provided in the bottom row.

Figure S4: MAD ablation experiments. Supplements Fig. 7 in the main paper.