HoneypotNet: Backdoor Attacks Against Model Extraction

Yixu Wang[1, 2*], **Tianle Gu**[2, 3], **Yan Teng**[2†], **Yingchun Wang**[2], **Xingjun Ma**[1, 2†]

[1]Shanghai Key Lab of Intell. Info. Processing, School of CS, Fudan University
[2]Shanghai Artificial Intelligence Laboratory
[3]Tsinghua University

Abstract

Model extraction attacks are one type of inference-time attacks that approximate the functionality and performance of a black-box victim model by launching a certain number of queries to the model and then leveraging the model's predictions to train a substitute model. These attacks pose severe security threats to production models and MLaaS platforms and could cause significant monetary losses to the model owners. A body of work has proposed to defend machine learning models against model extraction attacks, including both *active defense* methods that modify the model's outputs or increase the query overhead to avoid extraction and *passive defense* methods that detect malicious queries or leverage watermarks to perform post-verification. In this work, we introduce a new defense paradigm called **attack as defense** which modifies the model's output to be poisonous such that any malicious users that attempt to use the output to train a substitute model will be poisoned. To this end, we propose a novel lightweight backdoor attack method dubbed **HoneypotNet** that replaces the classification layer of the victim model with a honeypot layer and then fine-tunes the honeypot layer with a shadow model (to simulate model extraction) via bi-level optimization to modify its output to be poisonous while remaining the original performance. We empirically demonstrate on four commonly used benchmark datasets that HoneypotNet can inject backdoors into substitute models with a high success rate. The injected backdoor not only facilitates ownership verification but also disrupts the functionality of substitute models, serving as a significant deterrent to model extraction attacks.

Introduction

As the demand for integrating deep learning into daily tasks grows, Machine Learning as a Service (MLaaS) (Ribeiro, Grolinger, and Capretz 2015) has become a popular solution for deploying deep learning models across a wide range of applications. MLaaS platforms allow users to obtain prediction outputs through Application Programming Interfaces (APIs). However, research has revealed significant model leakage risks associated with MLaaS platforms due to model extraction attacks (Orekondy, Schiele, and Fritz 2019a; Pal et al. 2020; Yu et al. 2020; Zhou et al. 2020; Wang et al. 2022; Lin et al. 2023; Zhao et al. 2023; Karmakar and Basu 2023; Liu et al. 2024; Yuan et al. 2024). In a model extraction attack, an attacker approximates a black-box victim model by training a substitute model using a dataset constructed from queries to the victim model (Orekondy, Schiele, and Fritz 2019a). The attacker starts by querying the victim model with samples from an *attack dataset* that is either publicly available or synthesized. The predictions returned by the victim model serve as pseudo-labels to train the substitute model. After training, the substitute model often mimics the functionality of the victim model, enabling attackers to exploit it for various gains. Model extraction attacks thus pose significant threats to the intellectual property of deep learning models.

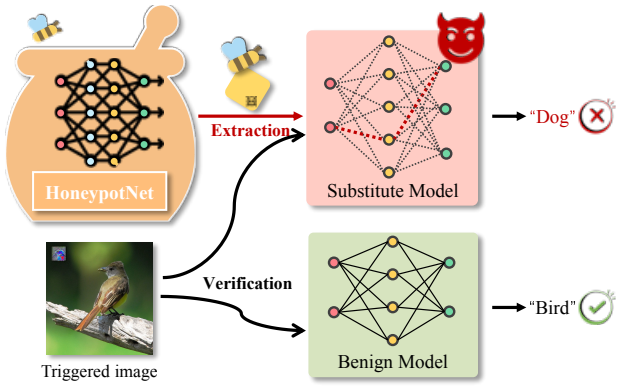

Figure 1: An illustration of our *HoneypotNet* defense.

Several methods have been proposed to defend deep learning models against model extraction attacks (Juuti et al. 2019; Dziedzic et al. 2022; Jia et al. 2021; Li et al. 2022; Lv et al. 2024; Orekondy, Schiele, and Fritz 2019b; Kariyappa and Qureshi 2020; Tang et al. 2024; Tan et al. 2023; Kesarwani et al. 2018). These defenses can be broadly categorized into two types: *active defense* and *passive defense*. *Passive defense* involves detecting potential attackers by monitoring user queries or using model watermarks for post-

*Work done during internship at Shanghai Artificial Intelligence Laboratory.

†Corresponding authors: <tengyan@pjlab.org.cn, xingjunma@fudan.edu.cn>

Defense Type	Method	Requirements		Capabilities		
		Computational Overhead	Logging User Queries	Hindering Extraction	Copyright Verification	Counter Attack
Passive defense	Extraction detection (2019; 2018)	Low	w/	✓	✗	✗
	Model watermarking (2021; 2022; 2024; 2023)	High	w/o	✗	✓	✗
Active defense	Proof-of-work (2022)	Low	w/	✓	✗	✗
	Prediction perturbation (2019b; 2020; 2024)	High	w/o	✓	✗	✗
Attack as defense	HoneypotNet (Ours)	Low	w/o	✗	✓	✓

Table 1: Comparison between different defense methods against model extraction attack. 'w/' and 'w/o' indicate with or without logging user query behavior, respectively. ✓and ✗denote whether a method has the listed functionality.

verification. However, these methods often rely on prior knowledge, making them less effective when such knowledge is unavailable. While watermarking can confirm model ownership, it must be integrated into the training process and does not guarantee that the watermark will be transferred to a substitute model. *Active defense* aims to prevent attackers from training effective substitute models by perturbing model outputs or increasing query overhead. However, these countermeasures may intensify the arms race between attackers and defenders, potentially leading to more sophisticated attacks.

In this paper, we introduce a novel defense paradigm called **attack as defense**. Unlike traditional active or passive defenses, this approach is more aggressive: it attacks the substitute model to disrupt its functionality and undermine the attacker's trust in it. To illustrate this paradigm, we present **HoneypotNet**, a lightweight backdoor attack method designed to protect image classification models from model extraction attacks. HoneypotNet is the first defense strategy that employs a backdoor attack on the substitute model, targeting any attackers who attempt to extract the victim model. As depicted in Figure 1, HoneypotNet replaces the output layer of the victim model with a *honeypot layer*, which is fine-tuned to produce poisonous probability vectors while preserving the model's original performance. If an attacker tries to extract the protected model and trains a substitute model using these poisoned probability vectors, the substitute model will be compromised and contain a backdoor. This approach allows the model owner to control the substitute model by exploiting the backdoor trigger, making it predict the backdoor class when activated.

The main challenge is to design a backdoor attack that injects a backdoor into the substitute model while ensuring the normal functionality of the victim model, and without explicitly adding backdoor triggers to the images, as the attacker will use their own clean images to train the substitute model. Inspired by adversarial examples (Szegedy et al. 2014; Goodfellow, Shlens, and Szegedy 2015), we propose using a specialized form of Universal Adversarial Perturbation (UAP) (Moosavi-Dezfooli et al. 2017) to address this challenge. As the adversarial vulnerability of deep learning models is inherent, UAPs can serve as poisoning-free triggers that do not require explicit injection. I.e., they function similarly to backdoors, where the presence of a UAP can activate a specific class. This leads us to fine-tune the honeypot layer to share the same adversarial vulnerability with a shadow model through a *Bi-Level Optimization (BLO)* framework. Here, the shadow model simulates the model extraction process. The shared adversarial vulnerability can then be transferred from the honeypot layer to any substitute model via its optimized poisonous probability vectors. We solve the BLO problem through alternating optimization, resulting in the honeypot layer and corresponding trigger after convergence. Experiments on four commonly used datasets show that our *HoneypotNet* defense achieves attack success rates between 56.99% and 92.35% on substitute models.

Related Work

Model Extraction Attack Model extraction attacks aim to extract (steal) a substitute model that mimics a victim model's functionality by querying its API. Papernot et al. (2017) first identified that an online model could be extracted by querying the black-box victim model multiple times. Existing model extraction techniques fall into two main categories: *data synthesis* and *data selection*. *Data synthesis* methods (Zhou et al. 2020; Kariyappa, Prakash, and Qureshi 2021; Lin et al. 2023; Liu et al. 2024; Yuan et al. 2024) use generative models, such as GANs (Goodfellow et al. 2014) or diffusion models (Ho, Jain, and Abbeel 2020), to create synthetic training data. However, these methods often require impractically large query volumes due to slow convergence. In contrast, *data selection* methods (Orekondy, Schiele, and Fritz 2019a; Wang and Lin 2022; Pal et al. 2020; Wang et al. 2022; Jindal et al. 2024; Zhao et al. 2024) choose informative samples from a pre-existing data pool. Techniques like reinforcement learning (e.g., KnockoffNets (Orekondy, Schiele, and Fritz 2019a)) or active learning (e.g., ActiveThief (Pal et al. 2020)) are used for this purpose. These approaches achieve high success rates with significantly fewer queries, making them a substantial threat in real-world scenarios. *Our work focuses on defenses against data selection-based extraction attacks.*

Model Extraction Defense The goal of model extraction defenses is to prevent or detect attempts to extract the victim model while ensuring legitimate user access. Existing defenses, summarized in Table 1, fall into four cate-

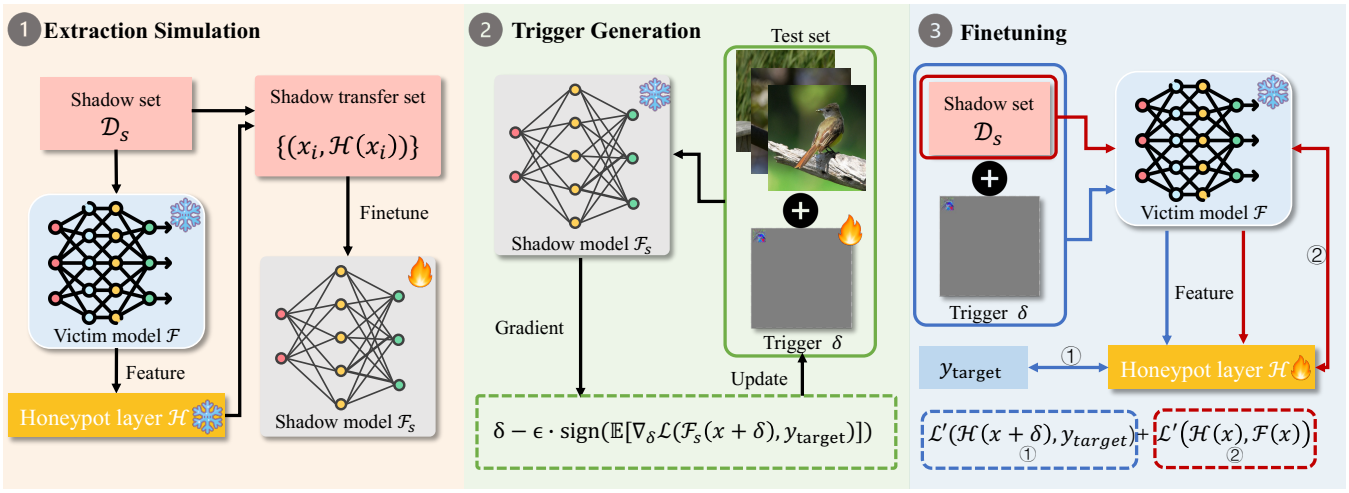

Figure 2: Overview of our HoneypotNet method. It replaces the classification layer of the victim model with a honeypot layer and finetunes the honeypot layer in three steps via bi-level optimization: 1) *extraction simulation*, which simulates the process of model extraction attacks with a shadow model; 2) *trigger generation*, which generates and updates the trigger on the shadow model; and 3) *finetuning*, which finetunes the honeypot layer with the trigger.

gories: The extraction detection (Juuti et al. 2019; Kesarwani et al. 2018) and proof-of-work (Dziedzic et al. 2022) methods log and monitor users' queries to detect malicious users. However, this logging behavior increases the risk of privacy leakage. Model watermarking techniques (Jia et al. 2021; Li et al. 2022; Lv et al. 2024; Tan et al. 2023) embed verifiable features into the model, but face limitations when applied to pre-trained models and offer minimal protection beyond ownership verification. The predictive perturbation method (Orekondy, Schiele, and Fritz 2019b; Kariyappa and Qureshi 2020; Tang et al. 2024) adds perturbations to the model's predictions to complicate the extraction process. However, this method is computationally expensive, as it requires calculating perturbations for each query sample. Additionally, it is vulnerable to advanced attacks that bypass these defenses using only hard labels (Wang et al. 2022; Sanyal, Addepalli, and Babu 2022; Yuan et al. 2024), underscoring the ongoing arms race between attackers and defenders. This paper introduces a novel defense paradigm termed **attack as defense**, which proactively targets the attacker rather than solely defending the model.

Backdoor Attack Backdoor attacks inject malicious behavior into deep neural networks (DNNs) by poisoning the training data with a trigger (Gu, Dolan-Gavitt, and Garg 2017). These attacks enable a backdoored model to operate normally on clean inputs while consistently predicting a target class when the trigger is present. Since their introduction in Chen et al. (2017), backdoor attacks have received substantial research attention (Gu, Dolan-Gavitt, and Garg 2017; Chen et al. 2021; Tang et al. 2020; Liu et al. 2019; Jha, Hayase, and Oh 2023; Chen et al. 2022; Rong et al. 2024). Existing poisoning-based backdoor attacks can be divided into two categories: standard dirty-image attacks (Gu, Dolan-Gavitt, and Garg 2017; Chen et al. 2017; Liu et al. 2019) and clean-image attacks (Jha, Hayase, and Oh 2023; Chen et al. 2022; Rong et al. 2024). Our proposed defense, **HoneypotNet**, is similar to clean-image attacks in that it injects a backdoor into the substitute model without altering the images. While previous research has examined clean-image attacks for multi-label classification (Chen et al. 2022), these methods rely on naturally occurring patterns and lack the use of specific triggers. FLIP (Jha, Hayase, and Oh 2023) overcomes this limitation by using an expert model and trajectory matching to select specific samples and their corresponding flipped labels. However, FLIP's need for access to the entire training dataset makes it unsuitable for our scenario.

Proposed Defense

Threat Model

We adopt a standard model extraction threat model where an attacker aims to extract a substitute model $\hat{\mathcal{F}}$ mimicking the functionality of a victim model $\mathcal{F} : [0,1]^d \mapsto \mathbb{R}^N$ using only black-box access. The attacker queries $\mathcal{F}$ with a chosen set of inputs to form a *transfer set* $\mathcal{D}_T = \boldsymbol{x}, \mathcal{F}(\boldsymbol{x})$. This transfer set, $\mathcal{D}_T$, is then used to train $\hat{\mathcal{F}}$, with the goal of achieving comparable accuracy: $\text{Acc}(\hat{\mathcal{F}}) \sim \text{Acc}(\mathcal{F})$. Our defense operates under the following assumptions: (1) Only the output $\mathcal{F}(\boldsymbol{x})$ can be modified, not the input $\boldsymbol{x}$; (2) Attacker queries are indistinguishable from legitimate user queries; (3) The victim model may be pre-trained, with its training data inaccessible. Therefore, our defense aims to inject backdoors into the attacker's substitute model without retraining the victim model or accessing its original training data.

The Honeypot Layer

We define the honeypot layer $\mathcal{H}$ as a fully connected layer, which takes the feature vector of the victim model as input and returns a probability vector: $\mathcal{H}(\boldsymbol{x}) = \boldsymbol{W} \cdot \mathcal{F}_{\text{feat}}(\boldsymbol{x}) + \boldsymbol{b}$,

where $\mathcal{F}_{\text{feat}}(\boldsymbol{x}) \in \mathbb{R}^m$ is the feature output (output at the last convolutional layer) of $\mathcal{F}$ on $\boldsymbol{x}$, $\boldsymbol{W} \in \mathbb{R}^{N \times m}$ is the weight matrix, and $\boldsymbol{b} \in \mathbb{R}^N$ is the bias vector. The honeypot layer $\mathcal{H}$ replaces the victim model's original classification layer to output poisonous prediction vectors. When an attacker uses the poisoned probability vectors to build the transfer set $\{\boldsymbol{x}, \mathcal{H}(\boldsymbol{x})\}$ and trains on it, the backdoor will be injected into the substitute model $\hat{\mathcal{F}}$. Formally, the effect $\mathcal{H}$ aims to achieve can be defined as:

$$\begin{aligned} &\underset{\theta_{\mathcal{H}}}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_{\text{test}}} \mathbb{I}(\hat{\mathcal{F}}(T(\boldsymbol{x})) = \boldsymbol{y}_{\text{target}}) \\ &\text{s.t.} \underset{\theta_{\mathcal{H}}}{\operatorname{argmax}}\, \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{D}_{\text{test}}} \mathbb{I}(\mathcal{H}(\boldsymbol{x}; \theta_{\mathcal{H}}) = \boldsymbol{y}), \\ &\text{where } \hat{\mathcal{F}} = \underset{\theta_{\hat{\mathcal{F}}}}{\operatorname{argmin}}\, \mathbb{E}_{(\boldsymbol{x}, \mathcal{H}(\boldsymbol{x})) \in \mathcal{D}_T} \left[\mathcal{L}\left(\hat{\mathcal{F}}(\boldsymbol{x}; \theta_{\hat{\mathcal{F}}}), \mathcal{H}(\boldsymbol{x}) \right) \right], \end{aligned} \tag{1}$$

where $T(\boldsymbol{x})$ is the operation that adds the trigger to $\boldsymbol{x}$, $\mathbb{I}$ is the indicator function, $\boldsymbol{y}_{\text{target}}$ is the predefined target backdoor label, $\mathcal{D}_{\text{test}}$ is the victim model's test set, and $\mathcal{L}$ is a loss function (e.g., cross-entropy, or Kullback–Leibler divergence (Kullback and Leibler 1951)).

Defending with a honeypot layer offers several advantages: (1) The honeypot layer has a small number of parameters, resulting in minimal computational overhead for finetuning; (2) It operates solely on the output features of the victim model, avoiding the need for retraining and making it suitable for large-scale pre-trained models; (3) The backdoor is introduced exclusively into the honeypot layer, ensuring that no additional security risks are posed to the victim model.

Finetuning the Honeypot Layer

Intuitively, the goal defined in Eq. (1) can be achieved through a backdoor attack. By embedding a backdoor into the honeypot layer, the attacker will inevitably extract this backdoor into their substitute model when replicating the functionality of the honeypot layer. However, as noted by Lv et al. (2024), model extraction attacks focus primarily on the key functions of the victim model, making it difficult to extract task-irrelevant backdoors into the substitute model. Additionally, since we cannot retrain the victim model, we are unable to use functionally relevant backdoors.

Inspired by the transferability of adversarial examples (Liu et al. 2017), we propose using a Universal Adversarial Perturbation (UAP) (Moosavi-Dezfooli et al. 2017) as an effective backdoor trigger. Our objective is to find a UAP (denoted as $\boldsymbol{\delta}$) that when applied to any input image $\boldsymbol{x}$, causes the substitute model $\hat{\mathcal{F}}$ to predict the target class $\boldsymbol{y}_{\text{target}}$:

$$\hat{\mathcal{F}}(T(\boldsymbol{x})) = \boldsymbol{y}_{\text{target}}; \quad \text{where } T(\boldsymbol{x}) = \text{clip}(\boldsymbol{x} + \boldsymbol{\delta}, 0, 1). \tag{2}$$

This perturbation $\boldsymbol{\delta}$ can be found through stochastic gradient descent (SGD) (Shafahi et al. 2020) with a gradient sign update, similar to the Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2015):

$$\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} - \epsilon \cdot \text{sign}(\mathbb{E}_{\boldsymbol{x}}[\nabla_{\boldsymbol{\delta}} \mathcal{L}(\hat{\mathcal{F}}(\boldsymbol{x} + \boldsymbol{\delta}), \boldsymbol{y}_{\text{target}})]), \tag{3}$$

where ϵ is the step size and $\text{sign}(\cdot)$ is the sign function.

As the defender, obtaining the substitute model is not feasible. Therefore, we introduce a ***shadow model*** $\mathcal{F}_s$ to approximate $\hat{\mathcal{F}}$ and a ***shadow set*** $\mathcal{D}_s$ to replace the attacker's transfer set $\mathcal{D}_T$. To minimize computational overhead, we use a lightweight model, such as ResNet18 (He et al. 2016), as $\mathcal{F}_s$. We employ $\mathcal{F}_s$ and the shadow set $\mathcal{D}_s$ to simulate the model extraction process. By solving Eq. (3) on $\mathcal{F}_s$, we obtain the UAP $\boldsymbol{\delta}$. This trigger is then used to fine-tune the Honeypot layer $\mathcal{H}$. This process is formulated as a Bi-Level Optimization (BLO) problem:

$$\underset{\theta_{\mathcal{H}}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_s} [\underbrace{\mathcal{L}(\mathcal{H}(\boldsymbol{x}), \mathcal{F}(\boldsymbol{x}))}_{\text{for normal functionality}} + \underbrace{\mathcal{L}(\mathcal{H}(\boldsymbol{x} + \boldsymbol{\delta}), \boldsymbol{y}_{\text{target}})}_{\text{for backdoor injection}}], \tag{4}$$

$$\begin{aligned} \text{s.t.} \quad &\underset{\theta_{\mathcal{F}_s}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_s} [\mathcal{L}(\mathcal{F}_s(\boldsymbol{x}), \mathcal{H}(\boldsymbol{x}))], \\ &\underset{\boldsymbol{\delta}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_v} [\mathcal{L}(\mathcal{F}_s(\boldsymbol{x} + \boldsymbol{\delta}), \boldsymbol{y}_{\text{target}})], \end{aligned} \tag{5}$$

where Eq. (4) is the upper level, Eq. (5) is the lower level, $\mathcal{D}_v$ is a small verification dataset related to the victim model's task, used to verify the backdoor attack success rate, and can be collected through an online search, and the sample classes should be evenly distributed as much as possible; and $\mathcal{L}$ is the cross-entropy loss.

Specifically, the BLO framework iteratively executes the following three steps: ***1) extraction simulation***, ***2) trigger generation***, and ***3) finetuning***. In the **extraction simulation** step, we randomly select n samples from the shadow dataset $\mathcal{D}_s$ for each iteration. We query the honeypot layer $\mathcal{H}$ to obtain predictions and then use sample-prediction pairs to train the shadow model $\mathcal{F}s$ for o epochs. This step simulates the process of an attacker obtaining a substitute model through model extraction. In the **trigger generation** step, the trigger $\boldsymbol{\delta}$ is updated on $\mathcal{F}_s$ according to the update rule defined in Eq. (3) for o epochs. To enhance concealment, a pre-defined mask $\boldsymbol{M}$ restricts the trigger to a specific location, and momentum is incorporated for improved optimization:

$$\begin{aligned} \boldsymbol{\delta}_i &= \alpha \cdot \boldsymbol{\delta}_{i-1} - (1 - \alpha) \cdot \epsilon \cdot \text{sign}(\mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_v}[\boldsymbol{g}(\boldsymbol{\delta}_{i-1})]), \\ \boldsymbol{g}(\boldsymbol{\delta}) &= \nabla_{\boldsymbol{\delta}} \mathcal{L}(\mathcal{F}_s(\boldsymbol{M} \odot \boldsymbol{x} + (1 - \boldsymbol{M}) \odot \boldsymbol{\delta}), \boldsymbol{y}_{\text{target}}), \end{aligned} \tag{6}$$

where $\boldsymbol{g}(\boldsymbol{\delta})$ denotes the gradient of loss $\mathcal{L}$ with respect to trigger $\boldsymbol{\delta}$, α is the momentum parameter and $\boldsymbol{M}$ is the binarized mask. In the **finetuning** step, the honeypot layer is finetuned according to Eq. (4), to maintain normal functionality while enhancing sensitivity to the trigger. This ensures that the extracted model inherits the backdoor vulnerability. In other words, the correlation between the UAP $\boldsymbol{\delta}$ and y_{target} is fine-tuned to be a normal functionality in the honeypot layer so that it will pass on to the extracted model. Since we cannot access the victim model's full training dataset, we also use the n samples randomly selected from $\mathcal{D}_s$ in the simulation step for finetuning. After m BLO iterations, the victim model with the honeypot layer will be deployed for its API service. We name the entire framework as ***HoneypotNet*** and outline its detailed procedure in Algorithm 1.

Algorithm 1: HoneypotNet

Input: Victim model $\mathcal{F}$, Shadow dataset $\mathcal{D}_s$.
Output: Honeypot Layer $\mathcal{H}$, trigger $\boldsymbol{\delta}$.

Initialize $\mathcal{H}$, $\mathcal{F}_s$, and δ.
for epoch in o **do**
  Select i samples from $\mathcal{D}_s$ and query $\mathcal{H}$
  for epoch in o **do** ▷ Extraction simulation
    $\mathcal{L} = \sum_{\boldsymbol{x} \in \mathcal{D}_s} \mathcal{L}'(\mathcal{F}_s(\boldsymbol{x}), \mathcal{H}(\boldsymbol{x}))$
    $\mathcal{F}_s \leftarrow \text{update}(\mathcal{F}_s, \mathcal{L})$
  end for
  for epoch in o **do** ▷ Trigger generation
    Update $\boldsymbol{\delta}$ according to Eq. (6)
  end for
  for epoch in o **do** ▷ Finetuning
    Calculate the loss $\mathcal{L}$ according to Eq. (5)
    $\mathcal{H} \leftarrow \text{update}(\mathcal{H}, \mathcal{L})$
  end for
end for

Ownership Verification and Reverse Attack

Each deployed Honeypot layer in the protected model is equipped with a unique, optimized trigger. This trigger has two main purposes: robust ownership verification and facilitating a reverse attack against any unauthorized substitute model. ***Ownership verification***: To determine if a suspect model has been extracted from the protected model, the defender can query it with samples embedded with the trigger but without specifying the target class. If the suspect model shows a classification accuracy Acc_v on the target class y_{target} that exceeds a predefined threshold (e.g., 10%), this suggests the presence of the backdoor. Such elevated accuracy provides strong evidence of model extraction and confirms ownership. ***Reverse attack***: Unlike existing watermarking techniques that rely on specific watermarked samples for verification, our approach offers a more aggressive response to confirmed model extraction. The trigger δ acts as a universal key that disrupts the functionality of the substitute model. By embedding δ into any input sample, the defender can force the substitute model to consistently predict the target backdoor class. This manipulation has serious consequences for the attacker. Unaware of the backdoor, the attacker risks deploying a compromised model that results in erroneous predictions and potential reputational damage. This effectively forces the attacker to abandon their substitute model, which serves as a powerful deterrent against model extraction.

Experiments

Experimental Setup

Victim and Shadow Models Following previous works (Orekondy, Schiele, and Fritz 2019a,b), the victim models we consider are ResNet34 (He et al. 2016) models trained on four datasets: CIFAR10, CIFAR100 (Krizhevsky, Hinton et al. 2009), Caltech256 (Griffin, Holub, and Perona 2007), and CUBS200 (Wah et al. 2011). The clean test accuracy of victim models are 91.56%, 71.57%, 77.11%, and 78.44%. Consistent with prior work (Orekondy, Schiele, and Fritz 2019a; Juuti et al. 2019; Pal et al. 2020), we initially assume the attacker employs the same architecture to train the substitute model. The impact of varying substitute architectures is also explored in our analysis. For the shadow model, we opt for a smaller model, i.e., ResNet18 (He et al. 2016), to minimize computational overhead.

Attack and Shadow Datasets We chose the ImageNet (Russakovsky et al. 2015) dataset as the attack dataset, which contains 1.2M images. We resize images to fit the input size of victim models. For the shadow dataset, we randomly select 5,000 images from the CC3M (Sharma et al. 2018) dataset due to its distinct distribution from ImageNet. This simulates a realistic scenario where the defender does not know what the attacker will use as the attack dataset.

Training and Extraction Configuration We perform BLO for 30 iterations, with each iteration comprising three steps: (1) Extraction Simulation: We train a ResNet18 for 5 epochs using SGD (momentum 0.9, learning rate 0.1, cosine annealing) on a transfer set generated by querying the honeypot layer. (2) Trigger Generation: We update the trigger for 5 epochs with momentum 0.9. (3) Finetuning: We finetune the honeypot layer for 5 epochs using SGD (momentum 0.9, learning rate 0.02, cosine annealing). For models with a small input image size (CIFAR10 and CIFAR100), we select a 6×6 square located 4 pixels away from the upper-left corner as the trigger location. For models with a larger input image size (Caltech256 and CUBS200), we choose a 28×28 square trigger at the same location. For simplicity, the last class is designated as the target class. Following previous work (Orekondy, Schiele, and Fritz 2019a,b; Pal et al. 2020), we train substitute models for 200 epochs using SGD (momentum 0.9, learning rate 0.02, cosine annealing).

Evaluation Metrics We employ three metrics: Clean Test Accuracy (Acc_c), Verification Test Accuracy (Acc_v), and Attack Success Rate (ASR). Acc_c measures substitute models' performance on clean test samples, reflecting defense's ability to preserve victim model utility while remaining undetectable to attackers. Acc_v assesses substitute models' accuracy on a set of triggered samples from the non-target classes on the target label. A high Acc_v indicates a successful ownership verification. ASR quantifies the success rate of defender in reverse attack substitute models by forcing it to predict the target label on any triggered input.

Extraction Methods and Baseline Defenses To evaluate the effectiveness of HoneypotNet, we apply five state-of-the-art extraction attacks, i.e., KnockoffNets (Orekondy, Schiele, and Fritz 2019a), ActiveThief (Entropy & k-Center) (Pal et al. 2020), SPSG (Zhao et al. 2024), and BlackBox Dissector (Wang et al. 2022). Notably, BlackBox Dissector method performs extraction under a hard-label setting, posing a unique challenge to our defense. We compare HoneypotNet with two baseline methods: no defense and DVBW(Li et al. 2023b), a defense method employing backdoor attacks for dataset ownership verification.

Main Results

Effectiveness of HonetpotNet Table 2 presents the results with a query budget of 30,000. The attack is successful,

Extraction Method	Model	CIFAR10			CIFAR100			CUBS200			Caltech256		
		Acc_c	Acc_v	ASR	Acc_c	Acc_v	ASR	Acc_c	Acc_v	ASR	Acc_c	Acc_v	ASR
KnockoffNets	No defense	82.96	1.99	12.47	52.66	0.41	0.90	61.25	0.05	0.14	73.25	0.14	0.47
	DVBW	82.32	3.38	12.50	49.87	0.02	0.91	64.52	0.29	0.35	73.11	0.17	0.26
	HoneypotNet	82.28	**54.88**	**59.35**	50.59	**85.61**	**85.71**	60.04	**78.38**	**78.31**	69.03	**79.04**	**79.13**
ActivteThief (Entropy)	No defense	82.08	0.78	10.21	51.28	1.17	1.50	65.21	0.16	0.43	74.28	0.11	0.42
	DVBW	81.69	1.53	9.39	50.52	1.98	2.36	61.06	0.29	0.67	74.92	0.41	0.93
	HoneypotNet	82.04	**52.29**	**56.99**	52.56	**74.09**	**74.35**	62.53	**83.39**	**83.22**	70.86	**77.29**	**77.43**
ActivteThief (k-Center)	No defense	83.31	1.03	11.38	52.23	0.17	0.89	66.17	0.03	0.26	76.83	0.15	0.34
	DVBW	82.20	3.90	10.32	52.23	0.31	0.54	63.19	0.39	0.44	74.06	0.23	0.77
	HoneypotNet	82.13	**64.00**	**67.49**	52.11	**74.48**	**74.63**	65.19	**80.20**	**80.27**	72.57	**80.62**	**80.80**
SPSG	No defense	85.47	0.36	10.31	53.12	1.23	1.37	63.18	0.12	0.42	71.74	0.09	0.41
	DVBW	83.13	1.79	12.20	51.18	0.77	0.97	60.42	0.77	0.89	69.91	0.82	0.95
	HoneypotNet	83.33	**62.93**	**66.12**	52.05	**76.92**	**77.11**	61.37	**83.56**	**83.51**	67.06	**77.91**	**77.88**
BlackBox Dissector	No defense	76.64	0.42	9.47	40.02	0.38	0.60	33.04	0.24	0.76	50.50	0.07	0.28
	DVBW	74.58	2.14	12.62	37.15	0.12	0.49	31.83	0.16	0.69	45.75	0.43	0.77
	HoneypotNet	74.97	**76.26**	**78.59**	38.81	**79.87**	**80.05**	30.50	**92.61**	**92.35**	47.95	**78.90**	**78.98**

Table 2: Effectiveness of HonetpotNet: the $Acc_c \uparrow (\%)$, $Acc_v \uparrow (\%)$, and ASR $\uparrow (\%)$ of extracted substitute model from different defense methods by five model extraction attacks under 30k queries.

as all five extracted substitute models exhibit a high Acc_c value. This proves that HoneypotNet does not harm the utility of models and does not alert the attacker. Compared to the negligible Acc_v values of undefended models and those protected by DVBW (Li et al. 2023b), HoneypotNet achieves significantly higher verification accuracy (52.29%-92.61%), indicating its effectiveness in verifying model ownership. Most importantly, HoneypotNet achieves consistently high ASR, ranging from 56.99% to 92.35%, demonstrating its ability to effectively inject backdoors into substitute models and enable powerful reverse attacks. Notably, our method proves remarkably effective even in a challenging scenario of hard labels, achieving a 78.59%-92.35% ASR on substitute models extracted by BlackBox Dissector. This underscores its practical utility.

Influence of Trigger Size Here, we test the impact of trigger size on the results with the KnockoffNets and CIFAR10. We test trigger sizes varying from 1×1 to 15×15 and report Acc_c and ASR of the victim, HoneypotNet, and substitute models in Figure 3. The trend suggests that as the trigger size increases, the ASR becomes higher, indicating a better attack effect. It is worth noting that a larger trigger will also impact the victim model, showing an abrupt increase in ASR when the trigger size is larger than 6. Interestingly, we find that the clean performance (Acc_c) of HoneypotNet increases with the increase of the trigger size. We believe this is because larger triggers have a stronger attack capability and thus are easier to learn without losing much of the original performance. Overall, this experiment suggests that one should balance the trigger size and protection effect in real-world scenarios to strike a better trade-off.

Influence of Substitute Model Architecture Contrary to the previous assumption that the victim and substitute models have same architectures, here we test more substitute architectures that are different from the victim. We take the KnockoffNets attack as an example and report the results in Table 3. Given that our backdoor trigger is based on the transferability of UAP, its effectiveness can vary depending on the model architecture, as adversarial transferability is sensitive to model architecture. For instance, the VGG16 model exhibits lower robustness, leading to higher attack success rates on various datasets, such as reaching 97.16% ASR on CIFAR10. The DenseNet121 model on the other hand has a slightly lower ASR. However, the lowest ASR achieved by our HoneypotNet is still above 51%, demonstrating its effectiveness even when extracted using different substitute models. Additionally, we believe that using a more complex shadow model or even an ensemble of multiple shadow models can further improve HoneypotNet.

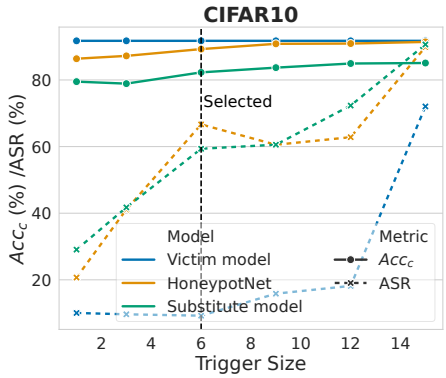

Figure 3: The impact of trigger size on the victim model, HoneypotNet, and the substitute model (extracted by the KnockoffNets attack under 30k queries) on CIFAR10.

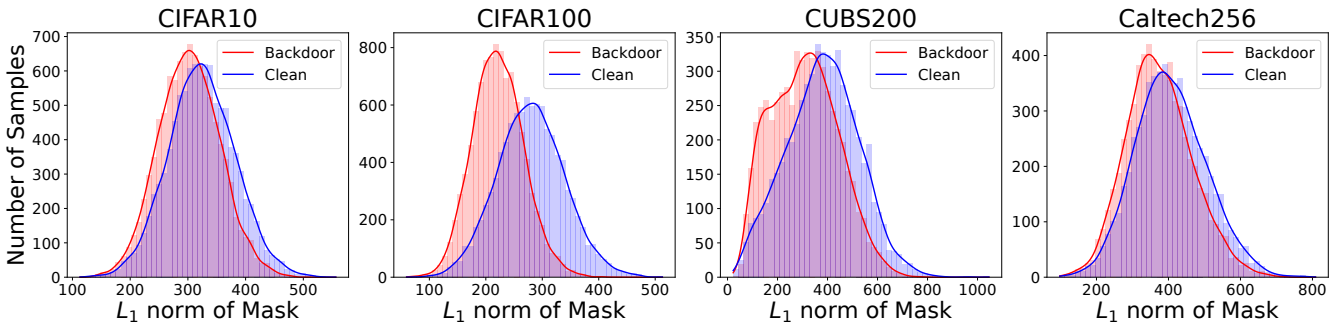

Figure 4: The L_1 norm distributions of the detected backdoor patterns for clean and backdoor samples by the backdoor detection method Cognitive Distillation (CD) (Huang et al. 2023).

Arc.	CIFAR10	CIFAR100	CUBS200	Caltech256
ResNet34	59.35	85.71	78.31	**79.13**
ResNet18	59.17	69.55	77.55	66.42
ResNet50	59.53	67.35	67.63	60.80
VGG16	**97.16**	**87.10**	**89.82**	62.17
DenseNet121	51.68	53.72	65.46	58.00

Table 3: The influence of different substitute model architectures on ASR (%). All substitute models were extracted by the KnockoffNets attack under 30k queries.

Evading Potential Backdoor Detection The attacker may leverage a backdoor detection method to detect whether there is a backdoor in the substitute model. To test this, here we consider a state-of-the-art backdoor detection method Cognitive Distillation (CD) (Huang et al. 2023) to test where it can detect the backdoor trigger pattern from the substitute model (extracted by KnockoffNets on CIFAR10). CD identifies backdoors by extracting minimal backdoor patterns (cognitive patterns) for a test image and comparing their L_1 norms between clean and potentially backdoored samples. A lower L_1 norm in backdoor samples suggests the presence of a shortcut pattern, indicating a backdoor. As shown in Figure 4, the L_1 norm distributions of clean vs. backdoor samples are very similar. This can be attributed to the inherent nature of our UAP-based trigger, which seamlessly integrates with the model's decision boundaries, mimicking natural features and evading detection by methods like CD that rely on identifying anomalous patterns.

Robustness against Backdoor Neuron Pruning To assess the robustness of our injected backdoor against possible neuron pruning by attackers, we evaluate the performance of HoneypotNet when countered by the Reconstructive Neuron Pruning (RNP) method (Li et al. 2023a). RNP aims to identify and prune backdoor neurons using only a small set of clean samples by leveraging an asymmetric unlearning-recovering process. We conduct experiments on the CIFAR10 dataset, utilizing the extracted substitute model from the KnockoffNets under 30k queries. We vary the number of clean samples used by RNP for defense and report the Acc_c and ASR at different steps of the RNP process in the Figure 5. Our results demonstrate that the ASR remains consistently high, with minimal impact from the varying defense data size. This highlights the robust nature of our backdoor injection, which seamlessly integrates the trigger into the model's normal functionality, making it difficult to be detected and pruned by defense mechanisms like RNP. This finding further underscores the effectiveness of HoneypotNet against sophisticated pruning-based defenses.

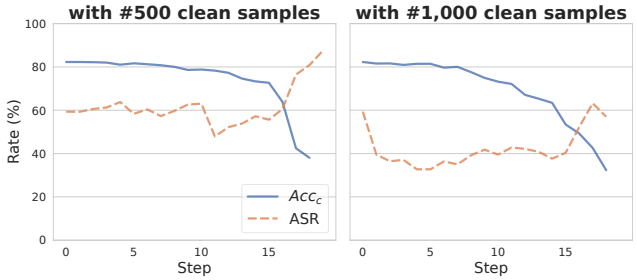

Figure 5: Robustness of HoneypotNet against Reconstructive Neuron Pruning (RNP) on the CIFAR10 dataset. The Acc_c and ASR are reported at different steps of the RNP process with 500 (left) and 1,000 (right) clean samples.

Conclusion

We introduce a novel defense paradigm called *attack as defense* which actively releases poisonous outputs to counteract and disrupt model extraction attacks. Unlike existing defense methods, *attack as defense* not only verifies model ownership but also hampers the functionality of substitute models. We present a specific implementation of this paradigm, **HoneypotNet**, which replaces the classification head of the victim model with a *honeypot layer* designed to generate both correct and poisonous probability vectors. The honeypot layer is fine-tuned using a shadow model and shadow dataset through bi-level optimization (BLO). Our empirical evaluation of HoneypotNet across four benchmark datasets, five model extraction attacks, and various substitute model architectures demonstrates its effectiveness. We hope our work will inspire further research into *attack as defense* strategies against model extraction attacks.

Acknowledgments

This work is in part supported by the National Key R&D Program of China (Grant No. 2022ZD0160103), the National Natural Science Foundation of China (Grant No. 62276067), and Shanghai Artificial Intelligence Laboratory.

References

Chen, H.; Fu, C.; Zhao, J.; and Koushanfar, F. 2021. Proflip: Targeted trojan attack with progressive bit flips. In *ICCV*.

Chen, K.; Lou, X.; Xu, G.; Li, J.; and Zhang, T. 2022. Clean-image backdoor: Attacking multi-label models with poisoned labels only. In *ICLR*.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Dziedzic, A.; Kaleem, M. A.; Lu, Y. S.; and Papernot, N. 2022. Increasing the cost of model extraction with calibrated proof of work. In *ICLR*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. In *NeurIPS*.

Huang, H.; Ma, X.; Erfani, S. M.; and Bailey, J. 2023. Distilling Cognitive Backdoor Patterns within an Image. In *ICLR*.

Jha, R.; Hayase, J.; and Oh, S. 2023. Label poisoning is all you need. *NeurIPS*.

Jia, H.; Choquette-Choo, C. A.; Chandrasekaran, V.; and Papernot, N. 2021. Entangled watermarks as a defense against model extraction. In *USENIX Security Symposium*.

Jindal, A.; Goyal, V.; Anand, S.; and Arora, C. 2024. Army of Thieves: Enhancing Black-Box Model Extraction via Ensemble based sample selection. In *WACV*.

Juuti, M.; Szyller, S.; Marchal, S.; and Asokan, N. 2019. PRADA: protecting against DNN model stealing attacks. In *IEEE EuroS&P*.

Kariyappa, S.; Prakash, A.; and Qureshi, M. 2021. MAZE: Data-Free Model Stealing Attack Using Zeroth-Order Gradient Estimation. In *CVPR*.

Kariyappa, S.; and Qureshi, M. K. 2020. Defending Against Model Stealing Attacks with Adaptive Misinformation. In *CVPR*.

Karmakar, P.; and Basu, D. 2023. Marich: A Query-efficient Distributionally Equivalent Model Extraction Attack using Public Data. *arXiv preprint arXiv:2302.08466*.

Kesarwani, M.; Mukhoty, B.; Arya, V.; and Mehta, S. 2018. Model extraction warning in mlaas paradigm. In *ACSAC*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kullback, S.; and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics*.

Li, Y.; Lyu, X.; Ma, X.; Koren, N.; Lyu, L.; Li, B.; and Jiang, Y.-G. 2023a. Reconstructive neuron pruning for backdoor defense. In *ICML*.

Li, Y.; Zhu, L.; Jia, X.; Jiang, Y.; Xia, S.-T.; and Cao, X. 2022. Defending against model stealing via verifying embedded external features. In *AAAI*.

Li, Y.; Zhu, M.; Yang, X.; Jiang, Y.; Wei, T.; and Xia, S.-T. 2023b. Black-box dataset ownership verification via backdoor watermarking. *IEEE TIFS*.

Lin, Z.; Xu, K.; Fang, C.; Zheng, H.; Ahmed Jaheezuddin, A.; and Shi, J. 2023. QUDA: Query-Limited Data-Free Model Extraction. In *ACM Asia CCS*.

Liu, Y.; Chen, X.; Liu, C.; and Song, D. 2017. Delving into transferable adversarial examples and black-box attacks. In *ICLR*.

Liu, Y.; Lee, W.-C.; Tao, G.; Ma, S.; Aafer, Y.; and Zhang, X. 2019. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *ACM SIGSAC CCS*.

Liu, Y.; Wen, R.; Backes, M.; and Zhang, Y. 2024. Efficient Data-Free Model Stealing with Label Diversity. *arXiv preprint arXiv:2404.00108*.

Lv, P.; Ma, H.; Chen, K.; Zhou, J.; Zhang, S.; Liang, R.; Zhu, S.; Li, P.; and Zhang, Y. 2024. MEA-Defender: A Robust Watermark against Model Extraction Attack. In *IEEE S&P*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *CVPR*.

Orekondy, T.; Schiele, B.; and Fritz, M. 2019a. Knockoff Nets: Stealing Functionality of Black-Box Models. In *CVPR*.

Orekondy, T.; Schiele, B.; and Fritz, M. 2019b. Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks. In *ICLR*.

Pal, S.; Gupta, Y.; Shukla, A.; Kanade, A.; Shevade, S.; and Ganapathy, V. 2020. ACTIVETHIEF: Model Extraction Using Active Learning and Unannotated Public Data. In *AAAI*.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *ACM AsiACCS*.

Ribeiro, M.; Grolinger, K.; and Capretz, M. A. 2015. Mlaas: Machine learning as a service. In *IEEE ICMLA*.

Rong, D.; Shen, S.; Fu, X.; Qian, P.; Chen, J.; He, Q.; Fu, X.; and Wang, W. 2024. Clean-image Backdoor Attacks. *arXiv preprint arXiv:2403.15010*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV*.

Sanyal, S.; Addepalli, S.; and Babu, R. V. 2022. Towards data-free model stealing in a hard label setting. In *CVPR*.

Shafahi, A.; Najibi, M.; Xu, Z.; Dickerson, J.; Davis, L. S.; and Goldstein, T. 2020. Universal adversarial training. In *AAAI*.

Sharma, P.; Ding, N.; Goodman, S.; and Soricut, R. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *ACL*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *ICLR*.

Tan, J.; Zhong, N.; Qian, Z.; Zhang, X.; and Li, S. 2023. Deep Neural Network Watermarking against Model Extraction Attack. In *ACM MM*.

Tang, M.; Dai, A.; DiValentin, L.; Ding, A.; Hass, A.; Gong, N. Z.; and Chen, Y. 2024. MODELGUARD: Information-Theoretic Defense Against Model Extraction Attacks. In *USENIX Security Symposium*.

Tang, R.; Du, M.; Liu, N.; Yang, F.; and Hu, X. 2020. An embarrassingly simple approach for trojan attack in deep neural networks. In *ACM SIGKDD*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset.

Wang, Y.; Li, J.; Liu, H.; Wang, Y.; Wu, Y.; Huang, F.; and Ji, R. 2022. Black-Box Dissector: Towards Erasing-based Hard-Label Model Stealing Attack. In *ECCV*.

Wang, Y.; and Lin, X. 2022. Enhance model stealing attack via label refining. In *ICSP*. IEEE.

Yu, H.; Yang, K.; Zhang, T.; Tsai, Y.-Y.; Ho, T.-Y.; and Jin, Y. 2020. CloudLeak: Large-Scale Deep Learning Models Stealing Through Adversarial Examples. In *NDSS*.

Yuan, X.; Chen, K.; Huang, W.; Zhang, J.; Zhang, W.; and Yu, N. 2024. Data-Free Hard-Label Robustness Stealing Attack. In *AAAI*.

Zhao, S.; Chen, K.; Hao, M.; Zhang, J.; Xu, G.; Li, H.; and Zhang, T. 2023. Extracting Cloud-based Model with Prior Knowledge. *arXiv preprint arXiv:2306.04192*.

Zhao, Y.; Deng, X.; Liu, Y.; Pei, X.; Xia, J.; and Chen, W. 2024. Fully Exploiting Every Real Sample: SuperPixel Sample Gradient Model Stealing. In *CVPR*.

Zhou, M.; Wu, J.; Liu, Y.; Liu, S.; and Zhu, C. 2020. DaST: Data-free Substitute Training for Adversarial Attacks. In *CVPR*.