Train to Defend: First Defense Against Cryptanalytic Neural Network Parameter Extraction Attacks

Ashley Kurian and Aydin Aysu
Department of Electrical and Computer Engineering
North Carolina State University
akurian@ncsu.edu, aaysu@ncsu.edu

Abstract

Neural networks are valuable intellectual property due to the significant computational cost, expert labor, and proprietary data involved in their development. Consequently, protecting their parameters is critical not only for maintaining a competitive advantage but also for enhancing the model's security and privacy. Prior works have demonstrated the growing capability of cryptanalytic attacks to scale to deeper models. In this paper, we present the first defense mechanism against cryptanalytic parameter extraction attacks. Our key insight is to eliminate the neuron uniqueness necessary for these attacks to succeed. We achieve this by a novel, extraction-aware training method. Specifically, we augment the standard loss function with an additional regularization term that minimizes the distance between neuron weights within a layer. Therefore, the proposed defense has zero area-delay overhead during inference. We evaluate the effectiveness of our approach in mitigating extraction attacks while analyzing the model accuracy across different architectures and datasets. When re-trained with the same model architecture, the results show that our defense incurs a marginal accuracy change of less than 1% with the modified loss function. Moreover, we present a theoretical framework to quantify the success probability of the attack. When tested comprehensively with prior attack settings, our defense demonstrated empirical success for sustained periods of extraction, whereas unprotected networks are extracted between 14 minutes to 4 hours.

1 Introduction

Neural networks hold significant value as their training process is computationally intensive, requires skilled labor, involves proprietary intellectual property, and demands extensive effort in dataset collection. Today, neural networks are ubiquitous, powering a wide range of applications due to their superior performance in various machine-learning tasks. The rise of machine learning as a service (MLaaS) has made it common for users to query models by providing input and receiving output, while model providers seek to obfuscate model parameters to protect their intellectual property. However, the increasing adoption of neural networks has also led to a rise in model-stealing attacks [25, 19, 17], which aim to create *high-fidelity* [11] replicas of the original models. These attacks target various components of the neural network, including hyperparameters [1, 26, 12] (such as layer types and configurations) and learned parameters [4, 9](trained weights and biases). The significant cost and effort savings compared to training a model from scratch make model theft an appealing strategy for adversaries. Moreover, model extraction eases other attacks, such as membership inference [22, 23] and input poisoning [5], further exacerbating security concerns.

Various attacks have demonstrated the feasibility of extracting neural network parameters through cryptanalytic techniques [4, 2, 21, 8]. These attacks treat parameter recovery as a structured mathematical problem, enabling the extraction of weights using a polynomial number of queries and within polynomial time with respect to model size on deep models [8]. Despite their effectiveness, cryptanalytic extraction attacks remain a major threat that has not been addressed to date, with their capabilities growing rapidly.

Preprint. Under review.

In this paper, we conduct the first in-depth analysis of the cryptanalytic extraction attack and identify key parameter conditions required for its success. Specifically, we observe that extracting weight magnitudes depends on the uniqueness of neuron weights within a layer—the less similar the weights, the more effective the attack. To mitigate this, we propose a training strategy that enforces weight similarity among intra-layer neurons, inherently enhancing the model's resistance to such attacks. This is achieved by incorporating a weight similarity constraint into the training loss function alongside the standard loss, which minimizes prediction error. However, enforcing similarity among intra-layer neuron weights can lead to a change in model accuracy. To address this, we explore various optimization strategies to ensure that the accuracy change remains minimal while maintaining a strong defense against parameter extraction. Furthermore, we evaluate our countermeasure across different neural network configurations used in prior end-to-end attack [8], including various datasets, architectures, training, and extraction seeds.

The main contributions of this paper include the following:

- We conduct a comprehensive analysis of cryptanalytic attacks, examining their underlying mechanisms and identifying the key parameter configurations that enhance their efficiency and likelihood of success. Understanding these crucial dependencies provides valuable insights into the fundamental weaknesses that adversaries exploit, laying the foundation for developing effective countermeasures against these attacks.
- In order to counteract cryptanalytic attacks, we propose an *extraction-aware training* strategy. This approach involves introducing a regularization term into the loss function that intentionally adjusts the model's parameters during training. The goal is to encourage parameter configurations that minimize the effectiveness of cryptanalytic attacks, making it significantly difficult for adversaries to extract sensitive model information successfully.
- A key advantage of our proposed defense is being zero-overhead at inference. The process is built into the training, and we show that re-training the same model hyperparameters with our technique can be sufficient to mitigate the attacks and has no run-time overheads.
- We analyze and implement various optimizations within our defense strategy to minimize the accuracy change caused by the inclusion of the defense mechanism while quantifying that the model remains robust and secure against cryptanalytic attacks. Our goal is to strike a balance between maintaining high model performance and enhancing resilience, thereby achieving an effective defense without compromising the model's accuracy.
- We evaluate our defense strategy across various neural network configurations that were attacked in prior works, including different datasets, model depths, numbers of neurons per layer, and training seeds. This evaluation allows us to assess the robustness and effectiveness of our defense under diverse conditions and configurations, ensuring its generalizability and reliability across a range of practical scenarios demonstrated in prior works.
- A major issue with prior attack evaluations was relying empirically on a set threshold (36 hours of attack runtime) for determining the attack success. To address this limitation, we propose a theoretical framework to assess attack success probability as a function of intra-layer neuron parameter similarity.

Our research demonstrates that extraction-aware training can effectively thwart cryptanalytic model parameter recovery attacks while preserving model accuracy with only a marginal change. By employing this lightweight defense strategy, we show that neural networks can be robustly protected from model stealing threats, offering a practical solution to safeguard sensitive model parameters without compromises in performance, memory footprint, or latency.

2 Background

2.1 Related work and our solution

Parameter extraction attacks on deep neural networks have been extensively studied over the past decades [7, 13, 25, 16, 20, 14]. These attacks seek to recover the weights and biases with sufficient numerical precision to achieve functional equivalence [11], meaning the extracted model should produce identical predictions to the original model for any input. Cryptanalytical parameter extraction attack exploits the fact that ReLU neural networks are piecewise linear functions, and thus queries at the points where the input to the activation is zero reveal information about the model parameters. The seminal cryptanalytic extraction attack shows the recovery of normalized values of weights and biases (*neuron signature*) with a polynomial number of queries with respect to model size [4]. However, their method required an additional exhaustive search to recover the signs of the weights, thereby introducing exponential time complexity to the attack. As a result, their evaluation was limited to shallow neural networks with a maximum of three layers. Martinez et al. [2] addressed the sign extraction limitation by introducing techniques to recover neuron signs, enabling a fully polynomial-time attack. They demonstrated sign extraction on deeper networks with up to eight layers. Foerster et al. [8] combined signature extraction and sign recovery to perform the first end-to-end parameter extraction attack. They introduced further optimizations to the sign extraction step, improving efficiency by 14.8×. These attacks assume access to confidence scores rather than just class labels for parameter extraction. More recently, cryptanalytic approaches have emerged that bypass this requirement and perform extraction using only the predicted labels [3, 6].

The advancements in neural network parameter extraction have raised significant concerns about protecting proprietary model values. However, no countermeasures have been developed to defend against these attacks. We examine the key assumptions and parameter configurations that contribute to the success of these attacks. Through this analysis, we observe that a fundamental assumption in signature extraction is that the intra-layer neuron parameters must be unique [4]. Moreover, certain neurons are classified as *hard neurons*, i.e., neurons whose signs are recovered with low confidence [8]. This occurs when it is impossible to craft an input perturbation that isolates the activity of the target neuron due to its similarity with other neurons. For such hard neurons in the previous layer, prior work propose recovering the next layer's signature by brute-forcing their signs. Their method performs parallel signature recovery across sign combinations, selecting the one that yields correct signatures for the next layer. The approach remains feasible for up to ten low-confidence neurons, beyond which it becomes computationally prohibitive [8].

Our goal in this paper is to leverage these insights and incorporate conditions into the training phase to make neurons *harder*. This is achieved by increasing intra-layer neuron parameter similarity, thereby making the model inherently resilient to cryptanalytic attacks. Specifically, we introduce a regularization term in the loss function that minimizes the distance between neuron parameters, while simultaneously optimizing for the primary learning task. While this approach strengthens model security, it impacts accuracy. To mitigate this trade-off, we explore several optimizations, including applying the defense only to the first layer, since protecting it can help safeguard the rest of the model in layer-wise attacks. We also evaluate strategies such as adjusting the strength of the regularization term to balance model accuracy and security. Finally, we assess the effectiveness and generalizability of our defense across different model architectures, datasets, training, and extraction seeds.

2.2 Threat model

We adopt the standard assumptions commonly used in neural network cryptanalytic parameter extraction attacks [4, 2, 21, 8, 3, 6]. The target model is a fully connected neural network that uses ReLU activations in the hidden layers and a linear activation in the output layer. The attacker has full knowledge of the target model's hyperparameters. The attack operates with high precision, capable of recovering parameters up to 32-bit floating point precision. The attacker is also granted the ability to input arbitrary data and observe the corresponding class labels or logits scores, effectively treating the model as a black-box system with the objective of reconstructing a functionally equivalent replica of the original network. The model can support any number of output classes. Side-channel attacks [10, 9] are considered out of scope, as they rely on auxiliary information such as power consumption or electromagnetic emissions, which may not be accessible in all application scenarios.

The overall attack setup is illustrated in Figure 1, which depicts the training and deployment phases of the model. As shown, the neural network is trained using a proprietary dataset and specific hyperparameters, including the model architecture—for example, three input nodes, two hidden layers with two nodes each, and four output classes <3,2,2,4>, with the mean squared error (MSE) loss function and the Adam optimizer. We follow the assumption that training is trusted. Once trained, the model, with its secret parameters, is deployed for inference. An attacker can then query the model to obtain class labels or logit scores to perform a cryptanalytic attack. The defender's objective is to secure the trained model parameters against such attacks.

The goal of the model provider is to build a robust neural network that remains resilient against cryptanalytic parameter extraction attacks at inference. From the defender's perspective, we assume full control over the training process, including the ability to modify the loss function. We assume

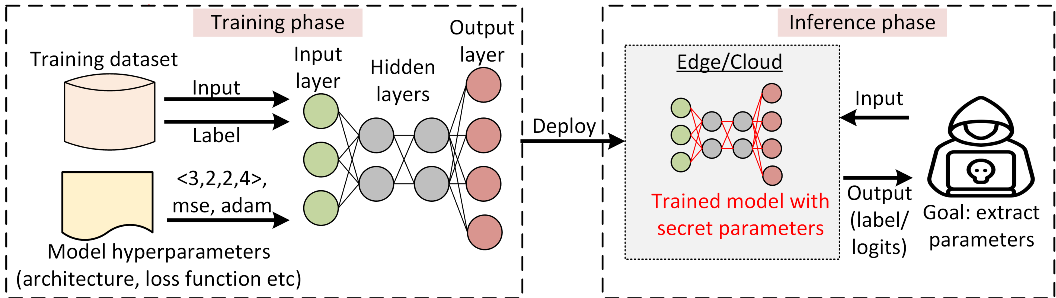


Figure 1: Schematic depicting how a deployed neural network becomes vulnerable to attack. A neural network is trained on a proprietary dataset and later deployed for inference. During the inference phase, an attacker queries the model to extract labels or logit scores for a cryptanalytic attack, while the defender aims to protect the model's secret parameters.

that training is trusted and not subject to model extraction attacks. Once deployed, the adversary can query the model and receive responses to execute cryptanalytic attacks during inference.

2.3 Cryptanalytic parameter extraction—A walkthrough

This section walks through the signature extraction technique [4] using a simple neural network for illustration. The attack recovers the weights of individual neurons by carefully probing how the output of the network changes in response to small input perturbations. The process begins by identifying critical points——input values where a neuron's ReLU output is zero following the terminology used in prior works [4, 2, 8]. Gradients are then computed around these points to capture directional information about the weights. By comparing the network's behavior when a neuron is active versus inactive, the attack isolates the influence of a single neuron and recovers its normalized parameters.

Consider a feedforward neural network with architecture $\langle 2, 2, 1\rangle$, as shown in Figure 2(a), where neuron *A* (η_A) has weights (a_1, a_2), neuron *B* (η_B) has weights (b_1, b_2) and the input vector *X*= (x_1, x_2). A *k*-deep neural network $f_\theta(x)$, parameterized by θ maps input space *X* to output space *Y*. With ReLU activation σ, the output is:

$$f_\theta(x) = c_1\sigma(a_1x_1 + a_2x_2) + c_2\sigma(b_1x_1 + b_2x_2) \tag{1}$$

Figure 2(b) shows the input space and corresponding critical hyperplanes of η_A and η_B, which represent the plane containing points where at least one of the ReLU inputs becomes zero. The critical points lie on the critical hyperplane. To recover the first-layer parameters, the attack traces the gradient flow from input coordinate *i* through the target neuron to the output. At each critical point x^*, the attack computes the directional derivatives of$f_\theta(x)$ by applying small positive and negative perturbations ϵ along the input axis e_i.

$$\alpha_i^+ = \frac{\partial f(x^* + \varepsilon e_i)}{\partial e_i}, \alpha_i^- = \frac{\partial f(x^* - \varepsilon e_i)}{\partial e_i} \tag{2}$$

where $e_i \in \mathbb{R}^{d_0=2}$, d_0 is the number of input neurons. The signature can then be recovered by computing $(\alpha_i^+ + \alpha_i^-)$ [4]. Assume we want to recover parameters of η_A using a critical point x^*. Suppose $x^* + \varepsilon e_i$ lies in a region where both η_A and η_B are active (represented using (+,+)), and $x^* - \varepsilon e_i$ lies in a region where η_A is inactive and η_B is active (represented using (-,+)). In this case:

$$f_\theta(x^* + \varepsilon e_1) = c_1(a_1(x_1 + \varepsilon e_1) + a_2x_2) + c_2(b_1(x_1 + \varepsilon e_1) + b_2x_2) \tag{3}$$

$$f_\theta(x^* - \varepsilon e_1) = c_1 \cdot 0 + c_2(b_1(x_1 - \varepsilon e_1) + b_2x_2) \tag{4}$$

The gradients are: $\alpha_1^+ = c_1a_1\varepsilon + c_2b_1\varepsilon$ and $\alpha_1^- = -c_2b_1\varepsilon$. Their sum $\alpha_1^+ + \alpha_1^- = c_1a_1\varepsilon$, isolates the first weight coordinate a_1 of η_A. Similarly, for direction 2, the sum $\alpha_2^+ + \alpha_2^- = c_1a_2\varepsilon$ reveals the second weight coordinate a_2. The normalized signature becomes:

$$\left(\frac{\alpha_1^+ + \alpha_1^-}{\alpha_1^+ + \alpha_1^-}, \frac{\alpha_2^+ + \alpha_2^-}{\alpha_1^+ + \alpha_1^-}\right) = \left(\frac{a_1}{a_1}, \frac{a_2}{a_1}\right) \tag{5}$$

This recovers the normalized weights of η_A. Notably, terms from other neurons cancel out (in this case, that of η_B), enabling isolated recovery of the target neuron's parameters. Repeating the process

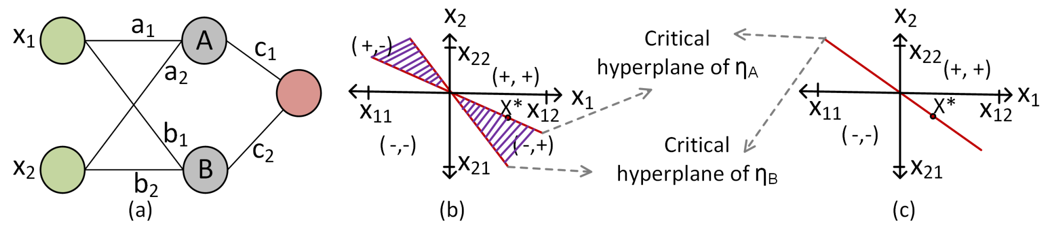


Figure 2: (a) neural network schematic with <2,2,1> configuration (b) input space showing the critical hyperplanes of η_A and η_B with distinct parameters (c) overlapping critical hyperplanes of η_A and η_B when their parameters are identical.

with a critical point for η_B allows recovery of its signature as well. See [4, 2, 21] for further details on signature and sign recovery. Attacks that rely on predicted labels adopt a similar strategy but begin by identifying critical hyperplanes through the detection of transition points—input values where the model's predicted class changes, thereby revealing the boundaries between decision regions [3].

3 Proposed defense against cryptanalytic parameter extraction

We propose the first countermeasure against cryptanalytic parameter extraction attacks. These attacks can target deep models trained on real-world datasets, requiring only black-box access in polynomial time and with a polynomial number of queries. Our defense addresses the main assumption underlying the success of these cryptanalytic attacks. The signature and sign extraction process is based on the distinctness of the neuron parameters, with the success rate of the attack increasing as the uniqueness between the neuron parameters increases. Our defense challenges this assumption by promoting similarity among neuron parameters during the training phase. This is achieved by adding a regularization term to the loss function that reduces the distance between neurons while still optimizing for the original loss function.

3.1 Why cryptanalytic attack fails by making neurons similar?

In this section, we provide a mathematical explanation of how making neuron parameters similar disrupts the cryptanalytic attack procedure, using the same example from Section 2.3. Consider the neural network shown in Figure 2(a). If the parameters of neurons η_A and η_B are the same—that is, $(a_1, a_2) = (b_1, b_2)$—their critical hyperplanes overlap. As a result, both neurons will be in the same state (either active or inactive) for any given input, as shown in Figure 2(c). Suppose we are trying to extract the signature of η_A and let x^* be a critical point. Assume that the input $x^* + \varepsilon e_i$ activates both neurons and the input $x^* - \varepsilon e_i$ deactivates both. Then, for direction 1:

$$f_\theta(x^* + \varepsilon e_1) = c_1(a_1(x_1 + \varepsilon e_1) + a_2 x_2) + c_2(b_1(x_1 + \varepsilon e_1) + b_2 x_2) \tag{6}$$

$$f_\theta(x^* - \varepsilon e_1) = c_1 \cdot 0 + c_2 \cdot 0 \tag{7}$$

The resulting gradients are: $\alpha_1^+ = c_1 a_1 \varepsilon + c_2 b_1 \varepsilon$ and $\alpha_1^- = 0$. The sum becomes: $\alpha_1^+ + \alpha_1^- = c_1 a_1 \varepsilon + c_2 b_1 \varepsilon$. This result includes the influence of both η_A and η_B, making it impossible to isolate the contribution of η_A for any given input perturbation. Therefore, the attack fails to recover the correct signature. Similarly, for direction 2: $\alpha_2^+ + \alpha_2^- = c_1 a_2 \varepsilon + c_2 b_2 \varepsilon$. The normalized signature is:

$$\left(\frac{\alpha_1^+ + \alpha_1^-}{\alpha_1^+ + \alpha_1^-},\ \frac{\alpha_2^+ + \alpha_2^-}{\alpha_1^+ + \alpha_1^-}\right) = \left(1,\ \frac{c_1 a_2 \varepsilon + c_2 b_2 \varepsilon}{c_1 a_1 \varepsilon + c_2 b_1 \varepsilon}\right) \tag{8}$$

This signature is incorrect because it mixes contributions from both neurons. The correct signature should be $(1, a_2/a_1)$, which reflects the behavior of only the target neuron (as derived in Section 2.3). Making neuron parameters similar causes them to activate or deactivate together for any input, preventing the attack from isolating a specific neuron and leading to its extraction.

3.2 Probability that a random input fails defense

A key limitation of prior attack evaluations is their reliance on an arbitrary empirical threshold (e.g., 36 hours of extraction) to define success. To address this, we introduce a theoretical framework that calculates the attack success probability as a function of the intra-layer neuron weight similarity. The key insight is that an attack fails when neurons are in the same state (either active or inactive) for any input, as it is not possible to isolate target neuron activity. However, if the parameters of the neurons differ, their critical hyperplanes diverge, and the attack succeeds. We quantify the probability of this success by calculating the area between the hyperplanes, where the neurons' states differ. This gives the probability of successfully extracting a single neuron pair. The total attack success probability is then obtained by identifying the pair of neurons with the greatest dissimilarity, representing the worst-case scenario.

Theorem 1. *The probability of extracting parameters of a neuron pair is the ratio between the measure of the input region lying between their corresponding hyperplanes and the total measure of the entire input space. This probability is is upper bounded by* $\frac{|K_1|+|K_2|+\cdots+|K_{N-1}|}{2} \cdot \frac{x_{12}^2+x_{11}^2}{(x_{12}-x_{11})^2}$ *where* $K_i = \frac{-a_i}{a_N} - \frac{-(a_i+\delta_i)}{a_N+\delta_N}$, *for* $i = 1, 2, \ldots, N-1$*;* N *denotes the input dimension,* a_i *refers to the* i*-th parameter of the neuron,* δ_i *is the* i*-th parameter difference between the neuron pair, and* x_{11} *and* x_{12} *are respectively the lower and upper limits of an input dimension.*

Proof. Consider the neural network in Figure 2(a). If the parameters of neurons η_A and η_B are identical, their critical hyperplanes overlap, as shown in Figure 2(c). In that case, both neurons always behave the same, and the defense works perfectly. However, when their parameters differ by (δ_1, δ_2)—meaning $b_1 = a_1 + \delta_1$ and $b_2 = a_2 + \delta_2$—their critical hyperplanes do not align. The equations for their critical hyperplanes are:

$$\text{For } \eta_A : a_1x_1 + a_2x_2 = 0 \tag{9}$$

$$\text{For } \eta_B : b_1x_1 + b_2x_2 = 0 \implies (a_1+\delta_1)x_1 + (a_2+\delta_2)x_2 = 0 \tag{10}$$

The shaded region in Figure 2(b) shows the area between these two hyperplanes, where the neurons behave differently. Inputs that fall in this area cause the attack to succeed. Let x_{11} and x_{12} be the lower and upper limits for the input x_1 and x_{21} and x_{22} for x_2. For INT8 inputs, these range from -128 to +127. For simplicity, we use x_{11} and x_{12} as the common lower and upper bounds for all input dimensions in the derivation, as each dimension shares the same input range.

$$\text{Area between hyperplanes of } \eta_A \text{ and } \eta_B = \int_{x_{11}}^{x_{12}} \left| \left(\frac{-a_1}{a_2} \cdot x_1\right) - \left(\frac{-(a_1+\delta_1)}{(a_2+\delta_2)} \cdot x_1\right) \right| dx_1 \tag{11}$$

$$= \int_{x_{11}}^{x_{12}} |K \cdot x_1|\, dx_1 = \frac{|K|}{2}\left(x_{12}^2 + x_{11}^2\right) \text{ where, } K = -\frac{a_1}{a_2} + \frac{a_1+\delta_1}{a_2+\delta_2} \tag{12}$$

$$\text{Total area of the input space} = \int_{x_{11}}^{x_{12}} \int_{x_{11}}^{x_{12}} dx_2\, dx_1 = (x_{12}-x_{11})(x_{12}-x_{11}) \tag{13}$$

$$\text{Attack success probability} = \frac{\text{Area between hyperplanes}}{\text{Total area}} = \frac{|K|}{2} \cdot \frac{x_{12}^2 + x_{11}^2}{(x_{12}-x_{11})^2} \tag{14}$$

This is the probability that a randomly chosen input will fall between the hyperplanes, meaning the neurons respond differently, and the attack succeeds. As the differences δ_1 and δ_2 increase, the two hyperplanes move further apart, increasing this probability. For an N-dimensional input space, due to the triangle inequality, the probability is upper bounded and can be computed as:

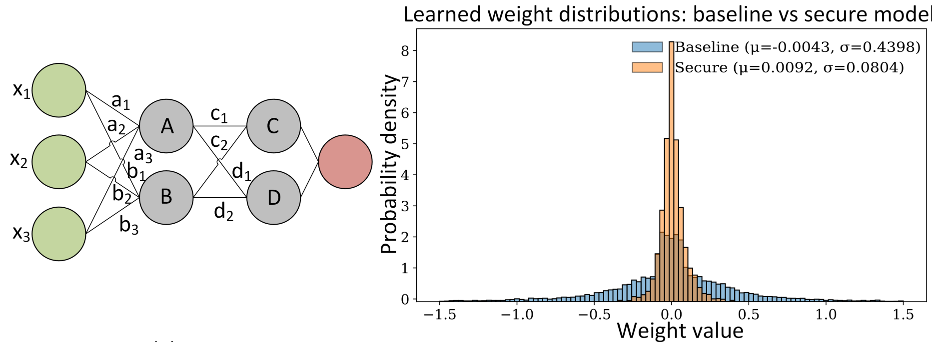
$$\text{Probability of attack success, } P \leq \frac{|K_1| + |K_2| + \cdots + |K_{N-1}|}{2} \cdot \frac{x_{12}^2 + x_{11}^2}{(x_{12}-x_{11})^2} \tag{15}$$

$$\text{where} \quad K_i = \left(\frac{-a_i}{a_N}\right) - \left(\frac{-(a_i+\delta_i)}{a_N+\delta_N}\right), \quad \text{for } i = 1, 2, \ldots, N-1 \qquad \blacksquare$$

This derivation assumes that the normal vectors of the two neurons' hyperplanes point in the same direction. In this case, the area between the hyperplanes (the shaded region in Figure 2(b)) represents where the neuron states differ, and thus where the attack succeeds. However, if the normal vectors point in opposite directions, then the region where the neurons differ in state flips—it becomes the region where they previously had the same state. In other words, the failure region becomes the unshaded area in Figure 2(b). So, in this case, the probability of attack success becomes $(1 - P)$. The direction of the normal vectors is captured by the angle θ between them:

$$\theta = \cos^{-1}\left(\frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}\right) \quad \text{where } \vec{a} = (a_1, a_2, \ldots, a_N),\ \vec{b} = (b_1, b_2, \ldots, b_N) \tag{16}$$

$$\text{Probability of attack success} = \begin{cases} P & \text{if } 0 \leq \theta \leq \frac{\pi}{2} \quad \text{(same direction)} \\ 1 - P & \text{if } \frac{\pi}{2} < \theta \leq \pi \quad \text{(opposite direction)} \end{cases} \tag{17}$$



(a) (b)

Figure 3: (a) neural network schematic with <3,2,2,1> configuration (b) comparison of weight distributions in the first dense layer of the secure and baseline models. The secure model shows a more concentrated distribution centered around zero with lower variance (0.08), while the baseline model exhibits a wider spread (variance 0.43). Only the range [–1.5, +1.5] is shown, which contains approximately 90% of the weights.

3.3 Extraction-aware neural network training

Adding a term to the loss function to increase similarity between neuron parameters can help defend against cryptanalytic attacks. We introduce a modified loss function:

$$\text{Loss function} = original\_loss + \lambda_{similarity} * total\_similarity\_loss \tag{18}$$

Here, $original\_loss$ refers to the standard function (such as MSE) that reduces the error between predicted and true labels. We introduce the additional term ($\lambda_{similarity} * total\_similarity\_loss$) to improve the security of the model. The factor $\lambda_{similarity}$ controls the impact of the similarity term, allowing a balance between accuracy and security——higher values increase security but may reduce accuracy. The $total\_similarity\_loss$ captures the overall similarity between neurons across all layers in the network:

$$total\_similarity\_loss = \Sigma_{\text{layer}=1}^{\text{number of layers}} layer\_wise\_similarity\_loss \tag{19}$$

$$layer\_wise\_similarity\_loss = \Sigma_{i=1}^{(\text{number of neurons}-1)} (p_i - p_{i+1})^2 \tag{20}$$

where p_i is the parameter of neuron i. The $layer\_wise\_similarity\_loss$ computes the pairwise difference between parameters of neurons within the same layer. This term uses L2 regularization to penalize large differences, making neurons more similar.

To illustrate, consider a neural network with architecture <3, 2, 2, 1> as shown in Figure 3(a). The $total\_similarity\_loss$ is the sum of the similarity losses for hidden layers 1 and 2:

$$total\_similarity\_loss = layer1\_loss + layer2\_loss$$

$layer1\_loss = (a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2$ and $layer2\_loss = (c_1 - d_1)^2 + (c_2 - d_2)^2$

In this example, the loss in each layer is computed based on the difference between the parameters of two neurons within that layer. For a layer with n neurons, there will be $\frac{n(n-1)}{2}$ pairwise comparisons. The goal is to make neurons behave similarly so that their outputs are the same for any input. This similarity prevents the attacker from isolating and targeting a specific neuron, as multiple neurons will respond in the same way.

3.4 Tuning our defense

The additional term $\lambda_{similarity} * total\_similarity\_loss$ is introduced in the loss function to enhance model security. However, this regularization must be carefully controlled to avoid compromising model accuracy. To mitigate accuracy degradation, we propose the following tuning strategies:

- Adjusting the regularization strength: By varying the impact factor $\lambda_{\text{similarity}}$, it is possible to control influence on accuracy while still making the model robust against attacks.
- Restricting defense to the first layer: Since the attack proceeds layer by layer, securing only the first layer can be sufficient. If the parameters of the first layer are protected, an attack on the subsequent layers could be automatically prevented.

Table 1: Cryptanalytic model parameter extraction performance

Model Information			**Signature [s]**		**Sign [s]**		**Queries**		**Secure Model Information**	
Model (Training Seed)	**Accuracy**	**Params**	**Mean**	**Var**	**Mean**	**Var**	**Mean**	**Var**	**Accuracy change (%)**	$\lambda_{similarity}$
Random784-8x2-1 (s1)	NA*	6280	$9.19 \cdot 10^2$	$9.6 \cdot 10^2$	54.69	36.6	$6.55 \cdot 10^5$	$2.53 \cdot 10^9$	NA	10^{-5}
Random784-16x2-1 (s1)	NA	12560	$1.37 \cdot 10^3$	$7.11 \cdot 10^2$	86.67	46.99	$1.31 \cdot 10^6$	$2.46 \cdot 10^7$	NA	10^{-4}
Random784-32x2-1 (s1)	NA	25120	$2.54 \cdot 10^3$	$4.29 \cdot 10^3$	$1.64 \cdot 10^2$	$1.7 \cdot 10^2$	$2.62 \cdot 10^6$	$1.41 \cdot 10^6$	NA	10^{-4}
Random784-64x2-1 (s1)	NA	50240	$5.43 \cdot 10^3$	$2.81 \cdot 10^5$	$2.78 \cdot 10^2$	$1.06 \cdot 10^3$	$5.24 \cdot 10^6$	$1.66 \cdot 10^{11}$	NA	10^{-1}
Random784-128x2-1 (s1)	NA	100480	$1.16 \cdot 10^4$	$7.61 \cdot 10^5$	$4.79 \cdot 10^2$	$1.07 \cdot 10^3$	$1.04 \cdot 10^7$	$6.57 \cdot 10^{11}$	NA	10^{-1}
Random784-128x2-1 (s2)	NA	100480	$1.38 \cdot 10^4$	$9.88 \cdot 10^4$	$5.54 \cdot 10^2$	8.12	$1.2 \cdot 10^7$	$3.2 \cdot 10^6$	NA	10^{-1}
MNIST784-8x2-1 (s2)	70.34	6280	$7.87 \cdot 10^2$	$5.74 \cdot 10^3$	48.54	26.52	$6.55 \cdot 10^5$	$7.74 \cdot 10^3$	-0.32	10^{-9}
MNIST784-16x2-1 (s2)	74.40	12560	$1.3 \cdot 10^3$	$3.93 \cdot 10^3$	95.71	0.29	$1.16 \cdot 10^6$	$2.27 \cdot 10^9$	$+0.74$	10^{-9}
MNIST784-32x2-1 (s2)	86.37	25120	$2.68 \cdot 10^3$	$5.48 \cdot 10^4$	$1.91 \cdot 10^2$	$2.81 \cdot 10^2$	$2.42 \cdot 10^6$	$2.8 \cdot 10^7$	$+0.83$	10^{-7}
MNIST784-64x2-1 (s2)	91.41	50240	$6.19 \cdot 10^3$	$1.56 \cdot 10^5$	$3.54 \cdot 10^2$	$4.96 \cdot 10^3$	$5.45 \cdot 10^6$	$4.06 \cdot 10^{10}$	-0.37	10^{-6}
MNIST784-64x2-1 (s1)	91.48	50240	$5.41 \cdot 10^3$	$5.28 \cdot 10^5$	$3.97 \cdot 10^2$	$2.34 \cdot 10^3$	$4.44 \cdot 10^6$	$5.93 \cdot 10^6$	-0.31	10^{-6}
MNIST784-16x8-1 (s2)	88.67	12560	$1.12 \cdot 10^4$	$1.95 \cdot 10^6$	80.37	0.41	$5.65 \cdot 10^6$	$8.91 \cdot 10^9$	-0.92	10^{-9}
MNIST784-16x3-1 (s1)	84.93	12560	$2.65 \cdot 10^3$	$1.99 \cdot 10^3$	$1.04 \cdot 10^2$	81.09	$1.91 \cdot 10^6$	$1.15 \cdot 10^5$	$+0.43$	10^{-9}

* NA: Accuracy not reported for random models as parameters are trained on random data

- Defending a subset of neuron pairs: Instead of enforcing similarity across all neurons in the first layer, the defense can target only a percentage of the neuron pairs. For example, a layer with eight neurons results in 28 pairwise similarity terms (i.e., $\frac{8(8-1)}{2}$). Defending 50% of the neurons involves enforcing similarity on 14 randomly selected pairs.

These optimization strategies help balance security and accuracy, ensuring that the defense mechanism does not adversely impact the model's performance.

4 Results

We evaluate our defense by measuring the model accuracy and attack run-time for parameter extraction. The models are trained using the modified loss function discussed in Section 3.3 to defend cryptanalytic parameter recovery attacks. To limit accuracy change, we apply optimizations such as restricting changes to the first layer parameters and adjusting $\lambda_{similarity}$. Table 1 shows results across different architectures, datasets, and training seeds. We follow the notation used in prior work [8]. For example, MNIST784_8x2_1(s2) model refers to an MNIST-trained network with an input layer of size 784, two hidden layers, each of size 8, and an output layer of size 1. We compare secure models to their baseline by reporting accuracy, attack runtime, and $\lambda_{similarity}$ that makes them secure. All training and extraction were done on a high-performance cluster with Intel's Xeon processors comprising 400 compute nodes and over 14,000 cores. On average, each node is provisioned with 128 GB of memory. We take our models from the papers that demonstrate attacks [8], and we use exactly all the hyperparameters evaluated in those papers. Models are trained on random data and MNIST, using two different training seeds (s1 = 42, s2 = 10). The extraction times for signatures and signs, number of queries, are reported as mean and variance across four extraction seeds (0, 10, 20, and 30). We consider a model to be secure if the attack fails after extended periods of extraction[1]. For example, the model MNIST784_16x8_1(s2), which was previously compromised in 3.5 hours, becomes secure with a $\lambda_{similarity}$ of 10^{-9}, while its accuracy drops by 0.92%. In some cases, the secure model even performs better. For instance, the MNIST784_16x3_1(s1) model shows a 0.43% increase in accuracy compared to its baseline version. The results show that for all the previously studied attack settings, our defense can secure the models through extraction-aware training with an accuracy change of less than 1% with zero-overhead inference.

Using the theoretical framework described in Section 3.2, we first compute the probability of attack success for models protected using our defense strategy, and then we empirically estimate the corresponding extraction time based on this probability. For example, the attack success probability for the MNIST784_16x8_1(s2) model is calculated based on the neuron pair in the first layer with the greatest parameter dissimilarity. The attack success probability drops from 0.74 for the baseline to 0.0017 for the secure version, a $435\times$ decrease. Figure 3(b) shows the weight distributions for the secure and baseline models. The baseline model's weights are spread across a wider range of [–2.5, +2.5], while the secure model's weights fall within [–0.2, +0.2] (Figure 3(b) shows only a range of [–1.5, +1.5], where 90% of the weights fall). To quantify this spread, the secure model has a lower weight variance of 0.08 compared to 0.43 in the baseline model, with weights more tightly clustered around zero. This indicates that the parameters in the secure model are more similar to each other, reducing the attack's ability to isolate neurons and extract their parameters individually.

[1]Prior work [8] uses 36 hours, whereas we use 48 hours as the extraction time limit.

5 Discussion

5.1 Where does the attack struggle for secure models?

The cryptanalytic parameter extraction on models secured by our defense is unsuccessful even after running the attack longer than the set threshold in prior papers [8]. The attack works by gathering critical points for each neuron to compute its signature. It then applies graph clustering to group critical points based on signature similarity. For the attack to succeed, each neuron should have a unique signature[2]; otherwise, it cannot specifically target individual neurons, as discussed in Section 3.1. The attack continues clustering until the number of clusters matches the number of neurons in the target layer. However, in secure models, neuron similarity leads to fewer clusters than expected, as some neurons share the same signature. The attack then collects more critical points by exploring multiple directions and repeating clustering. This process fails, which results in attack failure even after extended runtime. With our defense, the attack is still polynomial in time and number of queries. Our defense does not improve the theoretical security, but in practice, it increases the search complexity of the critical points per neuron.

5.2 Model extraction with partial parameter recovery

A question is whether an attacker can still recover the model by extracting only a few neurons in the first layer, or by randomly initializing the first layer and continuing the attack on subsequent layers. The answer is no. The attack on the following layers depends on having the correct output from the first layer, as it solves a system of equations based on those outputs. If the first layer is incorrect, the computed outputs are also wrong, leading to incorrect parameter recovery in the second layer. Worse, the errors accumulate with each deeper layer, causing the extraction to fail. Therefore, protecting only the first layer is enough to secure the entire model. Moreover, our proposed defense is not fundamentally limited to the first layer. It may be extended to future layers for other possible attacks that are not seeking *exact* high-fidelity capture with cryptanalysis.

5.3 Limitations

Our effort to secure models against cryptanalytic parameter extraction attacks comes at the expense of a marginal change (less than 1%) in accuracy. Additionally, as cryptanalytic attacks have not yet been demonstrated on CNNs or LLMs, our analysis focuses on the MLP network architectures evaluated in prior works. At present, these attacks apply only to MLPs with piecewise linear activations. These are limitations of the current attack (which is rapidly evolving), not of the defense. The defense itself has no such limitations and can scale as attacks advance, since it disrupts the key assumption underlying the attack——namely, neuron dissimilarity.

5.4 Other possible defenses

Other countermeasures such as parameter encapsulation, neural structure obfuscation, and injection of shortcuts or extra layers [27] can be employed to protect models. However, these approaches introduce computational overhead and increase the size of the ML library. Additional defenses against query-based model extraction reduce the effectiveness of such attacks through various strategies, including adding deceptive neurons [24], perturbing predictions by maximizing angular deviation between gradients to poison the attacker's training objective [18], or reprogramming the adversary with arbitrary behavior [15], but incur run-time or memory overhead. By contrast, our defense provides protection with zero overhead during inference.

6 Conclusion

Neural networks are valuable intellectual property that are vulnerable to parameter extraction attacks capable of recovering model parameters using cryptanalytic techniques in polynomial time and with a polynomial number of queries. In this paper, we present the first defense against such attacks. Our approach introduces an extraction-aware training method that adds a regularization term to the standard loss function. This term increases similarity between neuron parameters, making it difficult for the attack to isolate and recover individual neurons. The defense adds zero overhead during inference. We evaluate it across different network architectures, datasets, and training/extraction seeds. Results show a marginal change in accuracy of less than 1%, while protected models resist extraction even after extended periods compared to under four hours for unprotected models. We also provide a theoretical framework to analyze the probability of defense failure, offering further insight into the effectiveness of our method.

[2] *Zero weights* are excluded from this condition because neuron output remains constant regardless of the input.

References

[1] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 515–532, 2019.

[2] Isaac A Canales-Martínez, Jorge Chávez-Saab, Anna Hambitzer, Francisco Rodríguez-Henríquez, Nitin Satpute, and Adi Shamir. Polynomial Time Cryptanalytic Extraction of Neural Network Models. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2024.

[3] Nicholas Carlini, Jorge Chávez-Saab, Anna Hambitzer, Francisco Rodríguez-Henríquez, and Adi Shamir. Polynomial Time Cryptanalytic Extraction of Deep Neural Networks in the Hard-Label Setting. Cryptology ePrint Archive, Paper 2024/1580, 2024.

[4] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic Extraction of Neural Network Models. In *Annual international cryptology conference*, pages 189–218. Springer, 2020.

[5] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[6] Yi Chen, Xiaoyang Dong, Jian Guo, Yantian Shen, Anyu Wang, and Xiaoyun Wang. Hard-Label Cryptanalytic Extraction of Neural Network Models. Cryptology ePrint Archive, Paper 2024/1403, 2024.

[7] Charles Fefferman et al. Reconstructing a neural net from its output. *Revista Matemática Iberoamericana*, 10(3):507–556, 1994.

[8] Hanna Foerster, Robert Mullins, Ilia Shumailov, and Jamie Hayes. Beyond Slow Signs in High-fidelity Model Extraction. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 19496–19522. Curran Associates, Inc., 2024.

[9] Cheng Gongye, Yunsi Fei, and Thomas Wahl. Reverse-Engineering Deep Neural Networks Using Floating-Point Timing Side-Channels. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[10] Peter Horvath, Lukasz Chmielewski, Leo Weissbart, Lejla Batina, and Yuval Yarom. BarraCUDA: Edge GPUs do Leak DNN Weights. *arXiv preprint arXiv:2312.07783*, 2023.

[11] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High Accuracy and High Fidelity Extraction of Neural Networks. In *29th USENIX security symposium (USENIX Security 20)*, pages 1345–1362, 2020.

[12] Ashley Kurian, Anuj Dubey, Ferhat Yaman, and Aydin Aysu. TPUXtract: An Exhaustive Hyperparameter Extraction Framework. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(1):78–103, 2025.

[13] Daniel Lowd and Christopher Meek. Adversarial learning. In *Knowledge Discovery and Data Mining*, 2005.

[14] Flavio Martinelli, Berfin Simsek, Wulfram Gerstner, and Johanni Brea. Expand-and-Cluster: Parameter Recovery of Neural Networks, 2024.

[15] Mantas Mazeika, Bo Li, and David Forsyth. How to Steer Your Adversary: Targeted and Efficient Model Stealing Defenses with Gradient Redirection, 2022.

[16] Smitha Milli, Ludwig Schmidt, Anca D Dragan, and Moritz Hardt. Model Reconstruction from Model Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 1–9, 2019.

[17] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff Nets: Stealing Functionality of Black-Box Models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4954–4963, 2019.

[18] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks, 2020.

[19] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[20] David Rolnick and Konrad Kording. Reverse-Engineering Deep ReLU Networks. In *International conference on machine learning*, pages 8178–8187. PMLR, 2020.

[21] Adi Shamir, Isaac Canales-Martinez, Anna Hambitzer, Jorge Chavez-Saab, Francisco Rodrigez-Henriquez, and Nitin Satpute. Polynomial Time Cryptanalytic Extraction of Neural Network Models. *arXiv preprint arXiv:2310.08708*, 2023.

[22] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[23] Shubhi Shukla, Manaar Alam, Sarani Bhattacharya, Pabitra Mitra, and Debdeep Mukhopadhyay. "Whispering MLaaS": Exploiting Timing Channels to Compromise User Privacy in Deep Neural Networks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 587–613, 2023.

[24] Kálmán Szentannai, Jalal Al-Afandi, and András Horváth. Preventing Neural Network Weight Stealing via Network Obfuscation. In *Intelligent Computing: Proceedings of the 2020 Computing Conference, Volume 3*, pages 1–11. Springer, 2020.

[25] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[26] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. DeepEM: Deep Neural Networks Model Recovery through EM Side-Channel Information Leakage. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 209–218. IEEE, 2020.

[27] Mingyi Zhou, Xiang Gao, Jing Wu, John Grundy, Xiao Chen, Chunyang Chen, and Li Li. ModelObfuscator: Obfuscating Model Information to Protect Deployed ML-based Systems. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1005–1017, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim to develop an extraction-aware training method that defends against cryptanalytic attacks with zero overhead during inference, which is explained in Section 3. We also claim to empirically evaluate our defense across different neural network configurations; these results are shown in Table 1. We claim to provide a theoretical framework for computing the attack success probability, which is presented in Section 3.2.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Our extraction-aware training strategy causes an accuracy change of less than 1%. We quantify this extensively for all model settings used in prior works as shown in Table 1. The limitations of our proposed countermeasure are discussed in Section 5.3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: A theoretical proof of the attack success probability is provided in Section 3.2. The proof follows assumptions described in the threat model Section 2.2.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details of the machines used for experiments—a high-performance cluster with Intel Xeon processors. Information about the datasets, model configurations, training, and extraction seeds used to evaluate our defense is included in Section 4. We also report the impact factor ($\lambda_{\text{similarity}}$) required to secure each model. The code for reproducing the results is available at the GitHub link:
   [https://github.com/anonymous-123-code/anonymouscode]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a link to our GitHub code base: [https://github.com/anonymous-123-code/anonymouscode].

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All model settings, including configurations, training seeds, and extraction seeds, are reported in Table 1. Our codebase includes both the original (non-protected) and secure models, along with scripts to create new models using the same training setup in the previous paper [8]. All hyperparameters and optimizer choices are specified to enable full reproducibility. We adopt the same experimental settings, including hyperparameters and optimizer choices, as used in prior work [8].

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We follow the error reporting methodology provided in the previous work [8] by providing the mean, variances for parameter extraction time, and the number of queries for multiple test models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details of the machine setup used for experiments in Section 4. All experiments were conducted on a high-performance cluster equipped with Intel Xeon processors, comprising 400 compute nodes and over 14,000 cores. Each node is provisioned with an average of 128 GB of memory.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research complies with the NeurIPS Code of Ethics. It does not involve human subjects, personal data, or sensitive content, and uses only publicly available datasets. Our proposed defense enhances model security without enabling misuse.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper defends against parameter extraction attacks, which helps prevent model stealing and other threats like input poisoning and membership inference. This is important for protecting intellectual property and avoiding sensitive data leaks in critical areas like healthcare and defense. This is discussed in Section 1.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work uses only publicly available datasets (MNIST) and does not involve any models with proprietary intellectual property.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original authors of the attacks we defend against and clearly specify the datasets used (MNIST), which are publicly available.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide a link to our GitHub code base:
    [https://github.com/anonymous-123-code/anonymouscode]

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.