Geometric Critical Point Screening: Clustering-Free Cryptanalytic Extraction of Neural Network Models

Ming Duan and Peiyao Tang*

Information Engineering University, Zhengzhou, China
mdscience@sina.com, tpywswswws@foxmail.com

Abstract. Neural network model extraction has recently emerged as a critical security issue. In 2020, Carlini et al. categorized model extraction into signature extraction and sign extraction. In 2024, Canales-Martínez et al. proposed a polynomial-time sign extraction method. In 2026, Liu et al. achieved the first successful model extraction of 8-layer deep neural networks. However, existing signature extraction methods follow an inefficient compute-first, cluster-later paradigm: they first compute signatures for massive candidate critical points of unknown layer provenance, then separate points from different layers via clustering, which incurs prohibitive query and computational overhead.

This paper presents a geometric relationship-based critical point screening method. By searching for critical points on three coplanar parallel lines, we can rapidly separate critical points of first hidden layer neurons with minimal signature extraction, reducing the query complexity of signature extraction from $O(N \log N \cdot d_0)$ to $O(d_1 \cdot d_0)$. For neural networks where the input dimension exceeds the first hidden layer dimension, we can further achieve efficient screening of second hidden layer critical points by searching on three coplanar line segments within a fully activated space where all first hidden layer neurons are activated.

Geometric Critical Point Screening only requires computing signatures for a small number of non-target critical points. It offers advantages including low query cost and automatic validation of contaminated critical points. Experiments on a $784 - 8^{(8)} - 1$ network demonstrate that the time required for signature extraction of first and second hidden layer neurons is only 1.7% and 3.7% of existing methods, respectively, with query cost reduced to 3.2% and 0.1% of state-of-the-art approaches. Furthermore, this method is not limited to ReLU activation and can be extended to other piecewise linear activation functions, providing a fundamental and general lightweight approach for neural network model extraction.

Keywords: Deep Neural Networks · Neural Network Extraction · Geometric Screening · Cryptanalysis

* Corresponding authors.

1 Introduction

Deep Neural Networks (DNNs) are widely applied in computer vision, natural language processing, and other domains. However, training high-performance models requires collecting massive datasets and investing significant computational resources [20], making trained models valuable intellectual property. Most commercial models are deployed as black-box services [10], where users can only query inputs and obtain outputs through APIs without direct access to internal model parameters. If internal parameters can be recovered solely through API queries, it would pose a severe security threat to existing model deployment paradigms. This creates strong incentives for adversaries to steal commercially deployed models.

Research on model extraction attacks has a history of nearly three decades, and its development can be roughly divided into three phases:

The first phase is the theoretical foundation and early exploration stage (1994-2015). In 1994, Fefferman et al. [8] theoretically proved that the output function of a single-hidden-layer sigmoid neural network uniquely determines its architecture and weights (up to trivial equivalences of sign flips and permutations), laying the identifiability foundation for model extraction attacks. In 2005, Lowd and Meek [14] first proposed a reverse engineering algorithm for linear classifiers and successfully applied it to spam filtering systems, marking the transition of model extraction attacks from theory to practice. Research in this phase mainly focused on simple linear models and shallow neural networks, with limited practicality and extensibility of attack methods.

The second phase is the rise of practical attacks (2016-2019). In 2016, Tramèr et al. [19] published a landmark work at USENIX Security, demonstrating the first practical extraction attack against actually deployed machine learning APIs. They exploited confidence scores returned by APIs to successfully extract logistic regression, decision tree, and shallow neural network models. Subsequently, a series of research works emerged [16–18], extending the attack scope to more complex model types and stricter attack scenarios.

The third phase is the cryptanalytic extraction stage (2020-present). In 2020, Carlini et al. [4] proposed an attack method based on cryptanalytic principles at CRYPTO, analogizing ReLU neural networks to Substitution-Permutation Networks (SPNs) in block ciphers and decomposing the extraction process into two steps: signature extraction (recovering absolute values of weights) and sign extraction (determining weight signs). This method achieved functionally equivalent extraction of ReLU networks with up to 3 hidden layers and successfully extracted neural networks containing 100,000 parameters. This work achieved the first functionally equivalent extraction of ReLU networks [15] but had two key limitations: first, the time complexity of the sign extraction phase grew exponentially with the number of neurons, making it applicable only to extremely shallow networks; second, cumulative floating-point errors prevented signature extraction in deep networks. In 2023, Daniely et al. proposed a query-based polynomial-time algorithm for functionally equivalent extraction of ReLU neural networks with depth 2 and 3 [7].

In 2024, Canales-Martínez et al. [1] proposed two polynomial-time sign extraction algorithms at EUROCRYPT: the System of Equations (SOE) method and the Neuron Wiggle method, theoretically solving the exponential complexity problem of sign extraction. They reported successful extraction of a CIFAR-10 model with 8 hidden layers and 256 neurons per layer (totaling 1.2 million parameters). However, their experiments assumed perfect signature extraction and did not implement a complete end-to-end attack. Subsequent in-depth analysis by Foerster et al. [9] revealed a neglected core bottleneck: the signature extraction phase fails to reliably extract parameters beyond 3 hidden layers. Due to rank deficiency problems and deep noise interference, existing methods cannot reliably extract network parameters beyond 3 layers even with access to raw output values. Furthermore, Canales-Martínez's sign extraction method fails on so-called "low-confidence neurons," forcing Foerster et al. to reintroduce exponential exhaustive search to address this issue, making the polynomial-time promise not fully realizable.Neural network signature extraction under hard-label settings is even more challenging.A team from Tsinghua University proposed the first hard-label functionally equivalent extraction method for ReLU neural networks at ASIACRYPT 2024 [5]. Through a three-stage framework of dual-point localization, signature extraction, and sign extraction, they completed extraction of networks with hundreds of thousands of parameters within hours on a single-core CPU. Subsequently, teams from Google DeepMind and Stanford University achieved further theoretical breakthroughs at EUROCRYPT 2025 [3], proving that parameters of ReLU deep neural networks can be recovered in polynomial time under hard-label settings. They indirectly located critical hyperplanes by analyzing the geometric shape of decision boundaries, solving the core problem of difficulty in directly identifying critical points in hard-label environments. Follow-up work by the Technology Innovation Institute and the University of Campinas filled key gaps in the above research [2], addressing the issue that output layer parameters without ReLU activation cannot be extracted, and further proposed more efficient extraction strategies for contracting layers with decreasing neuron numbers, verifying the feasibility of complete deep neural network signature extraction on the CIFAR-10 dataset for the first time.

In 2026, Liu et al. [13] proposed a series of improved methods at EUROCRYPT, achieving the first end-to-end deep neural network extraction attack on 8-layer fully connected ReLU networks. They systematically solved the rank deficiency and deep noise problems in signature extraction and proposed a hybrid SOE+Wiggle sign extraction method, achieving the first end-to-end polynomial-time extraction of 8-hidden-layer ReLU networks. This milestone marked the entry of model extraction attacks into the deep network era, yet several critical issues remain unresolved:

First, existing methods still rely on a "multiple extraction-clustering consistency" strategy when processing the first two layers of neurons. To cluster neurons in each layer, at least $N \log N$ critical points must be collected to satisfy the coupon collector effect, where N is the total number of neurons in all hidden layers with ReLU activations. Moreover, due to activation issues of deep-

layer neurons, more critical points are often required to fully identify neuron layers and recover their signatures. For deep networks, this results in enormous query overhead. For example, in a $784 - 8^{(8)} - 1$ network, 3000-6000 critical points need to be collected and computed, and the vast majority of which belong to deep-layer neurons, causing severe resource waste in signature extraction of deeper-layer critical points.

Second, as the total number of neurons in a neural network increases, the frequency of "contaminated" critical points rises. When multiple linear regions exist near a critical point, the computed second derivative is a superposition of contributions from multiple neurons, leading to completely incorrect weight signatures. This further increases the number of critical points required for clustering, wasting additional query and computational resources.

In summary, despite significant progress in model extraction attack research, neural network model extraction remains a highly challenging open problem. All existing attacks face a common bottleneck: critical points from different layers are completely mixed in the input space, necessitating first computing signatures for numerous candidate critical points and then distinguishing layers through clustering. This approach incurs high query overhead and poor computational efficiency.

Our Contributions. In this paper, we conduct in-depth research on the above issues and propose a more efficient method for layer separation of critical points. Relying solely on geometric linear relationships, we can directly distinguish critical points of the first, second, and deep layers without clustering. This enables signature extraction of the first and second layers with minimal queries and significantly outperforms existing methods in terms of time. Based on this discovery, we construct the Geometric Critical Point Screening framework, which is not limited to specific network architectures, has fundamental generality, and can be directly embedded into all existing neural network model extraction pipelines. Experiments on a $784-8^{(8)}-1$ network show that our method reduces the query cost required for signature extraction of first- and second hidden layer neurons to below 3.2% and 0.1% of the previous state-of-the-art method, respectively. Our main contributions are as follows:

- **First Hidden Layer Screening Via Parallel Lines.** We propose a first hidden layer critical point screening method based on global linearity. Leveraging the global linearity of first hidden layer neuron hyperplanes, we search for critical points along three equidistant parallel lines in a two-dimensional plane and verify the midpoint property of critical points to directly separate first hidden layer from deep-layer critical points, eliminating the need for subsequent signature extraction and clustering operations.
- **Second Hidden Layer Screening Via Coplanar Line Segments.** We present a fast screening method for second hidden layer critical points based on a fully activated space where all first hidden layer neurons are activated. For neural networks satisfying input dimension greater than first hidden layer dimension, we find a fully activated space where all first hidden layer neurons

are activated. By constructing three coplanar line segments in this space and searching for critical points on these segments, we verify the collinearity property of critical points to directly separate second hidden layer from deep-layer critical points.
– **Generalization to ReLU-like Piecewise Linear Activation Functions.**We provide a fundamental, extensible, and cross-scenario model extraction method. Based on the properties of critical hyperplanes, this method is not only applicable to standard ReLU networks but also directly extendable to all piecewise linear activation functions such as Leaky ReLU and PReLU.

Theoretical analysis and experimental results demonstrate that our method significantly outperforms existing approaches in both query and time complexity. For the $784 - 8^{(8)} - 1$ network, the total query cost required for signature extraction of first and second hidden layer neurons is only 3.2% and 0.1% of existing methods, respectively, and this efficiency gap becomes more pronounced as network width increases.

Our code is available at https://github.com/1983321048/Geometric-Critical-Point-Screening.

Organization. The remainder of this paper is organized as follows: Section 2 introduces preliminaries; Section 3 details the first hidden layer neuron critical point screening method based on global linear relationships; Section 4 describes the fast screening method for second hidden layer neuron critical points based on local linearity; Section 5 presents experimental results and analysis; Section 6 concludes the paper and discusses future work.

2 Preliminaries

2.1 Basic Definitions and Notation

We follow the definitions and notation given in [4] to introduce several important formal definitions below. In this paper, we focus exclusively on the critical point screening problem during signature extraction. For signature extraction methods [4] and sign extraction methods [1], we refer readers to the respective original works for detailed protocols.

Definition 1 (k-Deep Neural Network [4]). *A k-layer neural network is a piecewise linear function $f : \mathcal{X} \to \mathcal{Y}$ composed of alternating linear layers f_i and activation functions σ:*

$$f = f_{k+1} \circ \sigma \circ f_k \circ \cdots \circ \sigma \circ f_1. \tag{1}$$

This paper only studies deep neural networks (DNNs) over the real number field, where $\mathcal{X} = \mathbb{R}^{d_0}$, $\mathcal{Y} = \mathbb{R}^{d_{k+1}}$, and only considers neural networks using ReLU as the activation function, $\sigma : x \mapsto \max(x, 0)$.

Definition 2 (Fully Connected Layer [4]). *The j-th layer of a neural network ($1 \leq j \leq k+1$) consists of an affine linear transformation whose input is the activation output of the previous layer and whose output is*

$$f_j(x) = A^{(j)}x + b^{(j)}. \tag{2}$$

where $A^{(j)} \in \mathbb{R}^{d_j \times d_{j-1}}$ is the weight matrix of the j-th layer, $b^{(j)} \in \mathbb{R}^{d_j}$ is the bias vector of the j-th layer, d_j is the number of neurons in the j-th layer, d_0 is the input layer dimension, and d_{k+1} is the output layer dimension.

Definition 3 (Total Number of Neurons [4]). *The total number of neurons N in a neural network is the sum of the number of neurons in all hidden layers, i.e.:*

$$N = \sum_{j=1}^{k} d_j. \tag{3}$$

The output layer does not contain ReLU activation functions and thus is not counted in the total number of neurons.

Definition 4 (Neural Network Architecture [4]). *The architecture of a neural network is completely determined by the following three parts:*

1. *The total number of hidden layers k of the network;*
2. *The dimension sequence of each layer $\{d_0, d_1, \ldots, d_k, d_{k+1}\}$;*
3. *Any additional constraints imposed on the weights $A^{(j)}$ and biases $b^{(j)}$ of each layer (such as sparsity constraints for convolutional layers).*

This paper uses the shorthand a-b-c to denote fully connected neural network architectures. For example, 784-32-1 *denotes a network with input dimension* 784*, a single hidden layer with 32 neurons, and output dimension* 1*. For consecutive hidden layers of the same width, exponential notation is used. For example,* $784-8^{(8)}-1$ *denotes a network with input dimension* 784*,* 8 *hidden layers each with* 8 *neurons, and output dimension 1.*

Definition 5 (Neuron State and Value [4]). *For any neuron η in the j-th layer ($1 \leq j \leq k$):*

1. *$V(\eta; x)$ denotes the input value of this neuron before ReLU activation when the input is x;*
2. *$S(\eta; x)$ denotes the activation state of this neuron, defined as:*

$$S(\eta; x) = \sigma(V(\eta; x)) = \max(V(\eta; x), 0). \tag{4}$$

If $S(\eta; x) > 0$, the neuron is said to be in the activated state; if $S(\eta; x) = 0$, the neuron is said to be in the inhibited state.

Definition 6 (Critical points [4]). *For any neuron η in the j-th layer, an input x is called a critical point of neuron η, denoted as $x \in W(\eta)$, if it satisfies*

$$V(\eta; x) = 0. \tag{5}$$

Critical points are the only nonlinear points in ReLU neural networks, where only the activation state of a neuron can flip.

Definition 7 (Critical Hyperplane [4]). *For any neuron $\eta_{j,i}$ in the j-th layer, the set of all its critical points $W(\eta_{j,i})$ is called the critical hyperplane of this neuron. According to Definition 6, critical hyperplanes of all layers can be uniformly expressed as*

$$A_i^{(j)} X_{j-1} + b_i^{(j)} = 0 \tag{6}$$

where X_{j-1} is the input to the j-th layer.

1. *When $j = 1$ (first hidden layer), $X_0 = x$ is the original input, so $W(\eta_{1,i})$ is a global linear hyperplane in the input space $\mathbb{R}^{d_0}$;*
2. *When $j \geq 2$ (deep hidden layers), $X_{j-1} = \sigma(f_{j-1}(X_{j-2}))$ is a piecewise linear function after ReLU activation of the previous layer, so $W(\eta_{j,i})$ is "bent" into a piecewise linear hyperplane by the nonlinear action of previous layers, whose shape depends on the activation states of all neurons in previous layers.*

2.2 Carlini et al.'s Attack Method

We briefly describe Carlini et al.'s attack method as a basis for comparison with our approach.

Random Search for Critical Points. A ReLU neural network can be regarded as a piecewise linear function. If no neuron undergoes an activation/inactivation state switch between two points (i.e., there are no critical points in the interval), then the model output in this interval strictly satisfies a linear relationship. Therefore, the output value at the midpoint equals the arithmetic mean of the output values at the two endpoints. If this linear relationship does not hold, it indicates that there is at least one critical point in the interval.

Randomly select an initial input point and a search direction to construct a search segment, compute the output values at the two endpoints of the segment; verify whether the midpoint output equals the average of the endpoint outputs. If not, there exists at least one critical point in the interval, and iteratively bisect the interval to narrow it down until a critical point meeting the required precision is located.

Signature Extraction and Compute First, Cluster Later Strategy. At the critical point of a single neuron, only the activation state of that neuron flips with a small input perturbation. At this point, computing the second derivative in the input direction directly corresponds to the product of the neuron's input weights and the output weights of subsequent layers, from which the proportional relationship of weights of neurons in the same layer can be derived.

Carlini et al.'s method adopts a "compute first, cluster later" strategy to distinguish critical points from different layers. The theoretical basis of this strategy is:

- For the j-th layer neurons currently being extracted, their critical points satisfy the same linear equation in the j-th layer input space, so the signatures computed from different critical points of the same neuron should be consistent;
- For all layers deeper than j, their critical hyperplanes, when mapped to the original input space, have been severely bent by the nonlinear effects of the first j layers. Therefore, the "pseudo-signatures" computed from critical points at different positions will exhibit a random distribution and almost never repeat.

After computing signatures for a large number of candidate critical points with unknown layer membership, signature values that occur significantly more frequently than expected by chance are more likely to belong to the j-th layer neurons currently being extracted.

However, this method requires signature extraction for all candidate critical points. Ideally, each critical point requires $2d_{j-1}$ queries, but due to errors, each critical point actually requires $4d_{j-1}$ queries, resulting in enormous query overhead for signature extraction. The geometric screening method proposed in this paper eliminates the need to compute signatures first to distinguish critical points from different layers, fundamentally solving this problem.

2.3 Adversarial Objectives and Resources

A model extraction attack involves two parties: an oracle $\mathcal{O}$ that returns model outputs $f_\theta(x)$, and an attacker who generates queries x to the oracle.

Definition 8. *A model signature extraction attack obtains oracle access to a parameterized function f_θ (a k-layer deep neural network in this paper) and the architecture information of function f, and returns a set of parameters $\hat{\theta}$. Its goal is to achieve functional equivalence , i.e., $f_\theta(x) = f_{\hat{\theta}}(x)$ for all inputs x.*

Throughout this paper, we use the $\hat{\cdot}$ notation to denote extracted parameters. For example, $\hat{\theta}$ denotes the weight parameters extracted from the original model θ.

The success of a model extraction attack is determined by two core dimensions: (a) the number of chosen-input queries issued to the model; (b) the total computational resources required for the attack.

Attack Assumptions. We make the following assumptions about the oracle $\mathcal{O}$ and the attacker's knowledge:

- **Known Architecture**: The attacker knows the complete architecture information of the target neural network, including network depth, number of neurons per layer, and inter-layer connection patterns.
- **Full Domain Input**: The attacker can select any real vector from the complete input space $\mathcal{X}$ as a query input.

- **Full Output Access**: The attacker can directly obtain the raw floating-point output of the model f without any post-processing (e.g., only returning the most likely class label without confidence scores).
- **Exact Computation**: The model f is defined and computed using 64-bit double-precision floating-point arithmetic.
- **ReLU Activation Function**: All hidden layers use the ReLU activation function $\sigma(\cdot)$.

3 First Hidden Layer Critical Points Screening

In model extraction attacks on deep ReLU neural networks, signature extraction first requires searching for critical points. When performing global random search on deep networks, the collected set contains critical points from all layers. To achieve signature extraction of the first layer, critical points of first hidden layer neurons must be identified. From Definition 7, the set of all critical points of a first hidden layer neuron forms a global linear hyperplane, while critical points of deep ($j \geq 2$) neurons correspond to hyperplanes that are bent by previous layer nonlinear transformations, exhibiting piecewise linear characteristics.

Definition 9. *The function that computes the first j layers of neural network f (including the j-th layer linear transformation f_j but excluding the activation function σ) is denoted as $f_{1..j}$. In particular, $f = f_{1..k+1}$.*

According to Definition 6, the critical point of the j-th layer neuron $\eta_{j,i}$ in a ReLU network satisfies

$$V(\eta_{j,i}; x) = A_i^{(j)} \sigma(f_{1..j-1}(x)) + b_i^{(j)} = 0. \tag{7}$$

For any critical point of a neuron $\eta_{l,m}$ in layer $l > j$, it satisfies

$$V(\eta_{l,m}; x) = A_m^{(l)} f_{j..l-1}(\sigma(f_{1..j-1}(x))) + b_m^{(l)} = 0. \tag{8}$$

Therefore, the critical hyperplane of the j-th layer neuron exhibits the characteristics of a linear hyperplane under the j-th layer input X_{j-1}. For neurons in layer l, since the first $l - j$ layers contain ReLU nonlinear transformations, $f_{j..l-1}(X_{j-1})$ is a piecewise linear function, so the critical point equation corresponds to a piecewise linear hyperplane. The hyperplanes of deep-layer neurons are cut into multiple linear segments by the hyperplanes of all previous neurons, presenting a "bent" morphology in the input space. Their intersection with any two-dimensional subplane is a polyline rather than a straight line. This essential difference constitutes the theoretical basis of our screening method.

3.1 Linear Screening Algorithm

For first hidden layer neurons ($j = 1$), the previous layer transformation degenerates to the identity map, and critical points satisfy

$$A_j^{(1)} x + b_j^{(1)} = 0. \tag{9}$$

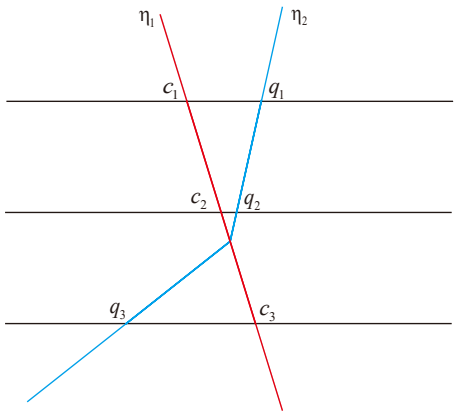

Fig. 1. For ease of understanding, we illustrate in a two-dimensional plane. The red line represents the critical hyperplane of the first hidden layer neuron η_1, and the blue line represents the critical hyperplane of the deeper-layer neuron η_2. Since the critical hyperplane of the deeper-layer neuron is bent by the critical hyperplanes of previous layer neurons, c_1, c_2, c_3 lie on the same straight line, while q_1, q_2, q_3 do not.

corresponding to a global linear hyperplane in the input space $\mathbb{R}^{d_0}$. The intersection of this hyperplane with any two-dimensional plane is a one-dimensional straight line, unless the plane is perpendicular to the hyperplane normal vector, in which case the hyperplane either does not intersect the two-dimensional plane or the intersection is the entire two-dimensional plane.

Based on this property, we propose the following linear screening algorithm (see Fig. 1):

Randomly select a reference point x_0 near the origin of the first hidden layer input space $\mathbb{R}^{d_0}$, randomly select two linearly independent unit vectors $u, v \in \mathbb{R}^{d_0}$, and choose a step size s (sufficiently large to ensure that the bending characteristics of deep hyperplanes are fully manifested) to construct three equidistant reference points

$$x_1 = x_0 - su, \quad x_2 = x_0, \quad x_3 = x_0 + su. \tag{10}$$

Search for critical points along the positive and negative directions of v starting from x_1, x_2, x_3 respectively, obtaining critical point sets C_1, C_2, C_3. Suppose $c_1 \in C_1, c_2 \in C_2, c_3 \in C_3$ are three critical points of the first hidden layer neuron η. Due to the global linear characteristic of the first hidden layer neuron hyperplane in $\mathbb{R}^{d_0}$, c_2 should be the midpoint of the line segment $\overline{c_1 c_3}$, i.e., satisfying

$$\|\boldsymbol{c_1} + \boldsymbol{c_3} - 2\boldsymbol{c_2}\|_2 < \tau. \tag{11}$$

where τ is a pre-defined small threshold for numerical error. In practical experiments,there is a negligible probability that critical points of deeper-layer neurons accidentally satisfy the above midpoint relationship. Therefore, we require

that the same set of parameters appears in at least two independent screening processes before we are confident that they correspond to critical points of first hidden layer neurons.

3.2 Query-Free Validation of Contaminated Critical Points

As the number of network neurons increases, multiple linear regions may exist simultaneously near a single critical point. At this point, computing the second derivative with a small step size will mix contributions from multiple neurons, leading to incorrect signature extraction. As the total number of neurons increases, the number of such critical points will increase, causing existing clustering methods to require more critical points for the same neuron.

Our method obtains three critical points (c_1, c_2, c_3) for the first hidden layer neuron η_j in one screening. After computing the $\hat{A}_j^{(1)}$ and $\hat{b}_j^{(1)}$ using one of the points c_1, we can directly validate using the other two points:

$$|\hat{A}_j^{(1)} c_2 + \hat{b}_j^{(1)}| < \tau, \quad |\hat{A}_j c_3 + \hat{b}_j^{(1)}| < \tau. \tag{12}$$

If the validation error exceeds the threshold, it indicates that the critical point is affected by multi-region interference and the calculation result is invalid. The entire process requires no additional model queries and can filter contaminated data in advance, further reducing query complexity.

3.3 Complexity and Advantage Analysis

In traditional extraction attacks, collecting sufficient first hidden layer neuron critical points relies on the coupon collector argument. Assuming the network has d_1 neurons in the first layer and critical points are uniformly distributed in the input space, it is necessary to collect S ($|S| \gg N \log N$) critical points to ensure at least 2 critical points per neuron, where N is the total number of hidden layer neurons in the network.

For each critical point, calculating its corresponding signatures requires $4d_0$ queries. Therefore, the signature extraction query complexity of traditional methods is $O(|S| \cdot d_0)$. Additionally, traditional methods require an additional clustering step with time complexity $O(|S|^2)$.

In contrast, our method directly leverages the global linearity of first hidden layer hyperplanes and can identify first hidden layer critical points based on geometric relationships. Theoretically, we only need to collect enough critical points and then compute signatures for d_1 critical points to achieve signature extraction of first hidden layer neurons, with a total query complexity of $O(d_1 \cdot d_0)$. In practice, the three critical point sets C_1, C_2 and C_3 each have a size of N. Instead of exhaustively enumerating all possible triplets (c_1, c_2, c_3) across the three sets, we adopt an optimized verification strategy: for any selected pair of points $c_1 \in C_1$ and $c_3 \in C_3$, we first compute the exact midpoint of the line segment $\overline{c_1 c_3}$, then only retrieve and verify the point $c_2 \in C_2$ that is closest to this midpoint. This allows us to complete the midpoint property verification with only

one check per (c_1, c_3) pair, eliminating the need to iterate over all elements in C_2. Since our method does not rely on clustering and only requires a constant-time verification for each (c_1, c_3) pair, our overall time complexity is approximately $O(N^2)$. When $N \gg d_1$ (which is common in modern deep networks), our method has significant advantages in both query and time complexity.

4 Second Hidden Layer Critical Points Screening

For neural networks satisfying $d_0 \geq d_1$ (input dimension greater than first hidden layer dimension), we can further extend the geometric screening method to critical points of second hidden layer neurons.

4.1 Existence Proof of Fully Activated Space

Theorem 1. *Assume the weight matrix $A^{(1)}$ of the first hidden layer has full row rank, for any neural network with input dimension greater than first hidden layer dimension ($d_0 \geq d_1$), there necessarily exists a non-empty, open, connected space $M_0 \subset \mathbb{R}^{d_0}$ such that for all $x \in M_0$, $V(\eta, x) \geq 0$ for every first hidden layer neuron η, i.e., $\sigma(f_1(x)) = f_1(x)$.*

Proof. The first hidden layer linear transformation can be expressed as

$$x_1 = A^{(1)}x_0 + b^{(1)} \tag{13}$$

where $x_0 \in \mathbb{R}^{d_0}$ is the original input and $x_1 \in \mathbb{R}^{d_1}$ is the pre-activation output vector of the first layer. Since $d_0 \geq d_1$ and the weight matrix $A^{(1)}$ is row-full-rank. Therefore, the linear system $A^{(1)}x_0 + b^{(1)} = x_1$ is consistent for any $x_1 \in \mathbb{R}^{d_1}$. We construct the open space $M_1 = [0, +\infty)^{d_1}$ in the pre-activation output space of the first hidden layer. For any $x_1 \in M_1$, there exists $x_0 \in \mathbb{R}^{d_0}$ such that $A^{(1)}x_0 + b^{(1)} = x_1$, hence $\sigma(A^{(1)}x_0 + b^{(1)}) = x_1$. The set of all such x_0 forms the fully activated space M_0, which is non-empty and connected.

We can construct a continuous region $M_1 = [0, +\infty)^{d_1}$ in the first hidden layer output space. By the existence of solutions to linear systems, for any $x_1 \in M_1$, we can easily compute $x_0 \in \mathbb{R}^{d_0}$ such that $\sigma(A^{(1)}x_0 + b^{(1)}) = x_1$.

The set of all such x_0 constitutes the fully activated space M_0 we seek. Clearly, for any $x \in M_0$, $V(\eta, x) \geq 0$, and the ReLU function degenerates to the identity map. Thus, the fully activated space M_0 is non-empty and continuous. □

Within this fully activated space M_0, the composite function of the first two layers degenerates to a linear function

$$f_{1..2}(x) = A^{(2)}(A^{(1)}x + b^{(1)}) + b^{(2)}. \tag{14}$$

Therefore, the critical hyperplanes of second hidden layer neurons remain linear within M_0, and their intersection with any two-dimensional subplane is a straight

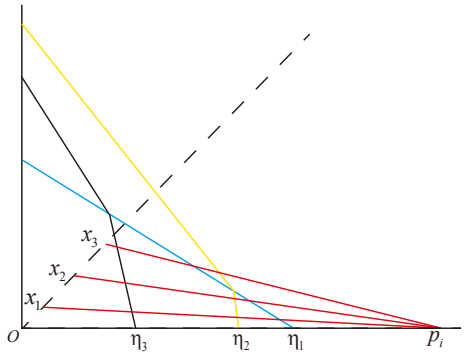

Fig. 2. For ease of understanding, we illustrate in a two-dimensional plane. We select three collinear points x_1, x_2, x_3 in the fully activated space M_0 of the second hidden layer input space, and connect them to a point p_i on a coordinate axis. We perform critical point search along these three line segments (red lines). The blue line represents the critical hyperplane of second hidden layer neuron η_1, the yellow line represents the critical hyperplane of deeper-layer neuron η_2, and the black line represents the critical hyperplane of deeper-layer neuron η_3. Although deeper-layer hyperplanes bend when crossing previous layer hyperplanes, some may still be misclassified as second hidden layer neurons. In this figure, η_2 is successfully filtered out, but η_3 is misclassified as a second hidden layer neuron and requires further screening.

line. For neurons in layer j ($j \geq 3$), however, their critical hyperplanes are still bent by the ReLU nonlinearity of the second layer even within M_0, exhibiting piecewise linear characteristics with polyline intersections with two-dimensional subplanes.

Based on this property, we propose the following linear screening algorithm (see Fig. 2):

4.2 Second Hidden Layer Critical Point Screening Algorithm

Proportional Scaling of First Hidden Layer Signature. The first hidden layer neuron parameters extracted via cryptanalytic attacks inherently suffer from ambiguity up to a positive scalar multiple. While this ambiguity does not affect functional equivalence, we perform critical point search at fixed locations. Such scale inconsistencies degrade the precision of critical point localization of critical point search and reduce the precision of signature extraction.

Using the first hidden layer critical points screened previously, we can complete proportional scaling of parameters: for each first hidden layer neuron η_i, select any corresponding critical point c_i, and compute the magnitude of the projection of the vector from the origin to c_i onto the extracted weight $\hat{A}_i^{(1)}$:

$$s_i = \frac{|\hat{A}_i^{(1)} c_i|}{\|\hat{A}_i^{(1)}\|_2}. \tag{15}$$

Compute the scaling factor $\alpha_i = \frac{s_i}{|\hat{b}_i^{(1)}|}$, and complete scaling by multiplying weights and biases by α_i:

$$A_i^{(1)} = \alpha_i \hat{A}_i^{(1)}, \quad b_i^{(1)} = \alpha_i \hat{b}_i^{(1)}. \tag{16}$$

After scaling, the extracted first hidden layer parameters are unified in scale with the true first hidden layer parameters. At this point, the neuron output value $A_i^{(1)}x + b_i^{(1)}$ equals the signed distance from input point x to the neuron's critical hyperplane.

Let $\mathbf{1}_d \in \mathbb{R}^d$ denote the d-dimensional all-ones vector, i.e., $\mathbf{1}_d = [1, 1, \ldots, 1]^T$; Let $\mathbf{e}_i \in \mathbb{R}^d$ denote the d-dimensional standard unit vector with 1 in the i-th component and 0 elsewhere, i.e., $\mathbf{e}_i = [0, \ldots, 0, \underset{i}{1}, 0, \ldots, 0]^T$.

Compute $x_1, x_2, x_3 \in \mathbb{R}^{d_0}$ such that:

$$A^{(1)}x_1 + b^{(1)} = t\mathbf{1}_{d_1}, \quad A^{(1)}x_2 + b^{(1)} = 2t\mathbf{1}_{d_1}, \quad A^{(1)}x_3 + b^{(1)} = 3t\mathbf{1}_{d_1}.$$

where t is typically chosen as a small positive value,Compute $p_i \in \mathbb{R}^{d_0}$ such that $A^{(1)}p_i + b^{(1)} = k\mathbf{e}_i$, k is typically chosen as a sufficiently large positive constant.

Since the first hidden layer outputs of points p_i, x_1, x_2, x_3 all lie in M_1, the line segments $\overline{p_i x_1}, \overline{p_i x_2}, \overline{p_i x_3}$ lie within a two-dimensional plane in M_0. The critical hyperplanes of second hidden layer neurons remain linear within M_0. Based on this property, we search for critical points along these specific line segments to obtain critical point sets C_1, C_2, C_3.

For critical points $c_1 \in C_1, c_2 \in C_2, c_3 \in C_3$ belonging to the same second hidden layer neuron, c_1, c_2, c_3 must be collinear, i.e.:

$$|\|\overrightarrow{c_1c_3}\| - (\|\overrightarrow{c_1c_2}\| + \|\overrightarrow{c_2c_3}\|)| < \tau. \tag{17}$$

We can filter out a large number of deeper-layer critical points using this method.

Secondary Validation of Deep-Layer Critical Points. A small number of deep-layer critical points may accidentally satisfy the above collinearity relationship. Therefore, we compute signatures $\hat{w}, \hat{b}$ for each triplet (c_1, c_2, c_3), then randomly generate $z_1 \in M_1$ satisfying $\hat{w}z_1 + \hat{b} = 0$, and compute $x_0 \in \mathbb{R}^{d_0}$ such that $\sigma(A^{(1)}x_0 + b^{(1)}) = z_1$.

Compute $\hat{w}' \in \mathbb{R}^{d_0}$ such that $A^{(1)}\hat{w}' = \hat{w}$, then calculate the second-order derivative of f at x_0 along direction $\hat{w}'$. If

$$\left| \frac{\partial^2 f(x_0)}{\partial(\hat{w}')^2} \right| < \tau, \tag{18}$$

then x_0 is not a critical point, and we discard (c_1, c_2, c_3) as deep-layer critical points.

In practice, due to numerical errors, some second hidden layer critical points may be falsely rejected. Therefore, we typically randomly generate 20 points satisfying the above requirements. If at least 15 of these points are verified as

critical points, we conclude that the triplet corresponds to a second hidden layer neuron. As the number of second hidden layer neurons increases, we expect this method to perform even better, as the bending of deeper-layer hyperplanes by second hidden layer hyperplanes becomes more pronounced.

4.3 Complexity and Advantage Analysis

Taking the $784 - 8^{(8)} - 1$ network as an example, existing methods require collecting K critical points when processing the second layer (Liu et al. used 3000-6000 critical points in their experiments), with each critical point requiring $4 \times 8 = 32$ queries. Since randomly searched critical points correspond to only partially activated first hidden layer neurons, additional large numbers of critical points are needed to recover all weights.

Our method significantly reduces the signature extraction overhead for deep-layer critical points. Through geometric screening within the fully activated space, we can directly separate second hidden layer critical points from candidate points. Theoretically, we only need to compute signatures for $d_2 = 8$ second hidden layer critical points; considering a small number of false positives from deep-layer neurons, in practice we only need to compute signatures for about 50 candidate points.

Furthermore, critical points obtained within M_0 correspond to fully activated first hidden layer neurons, so critical points in this region can recover all weights in one pass, greatly reducing signature extraction overhead. Experimental results show that the total query cost is approximately $2^{14.48}$, only 0.1% of existing methods.

Note that this method only requires the input dimension to be greater than the first hidden layer dimension, a condition satisfied by the vast majority of practically deployed deep learning models. Combined with the first hidden layer screening method from the previous section, we can achieve efficient extraction of the first two layers in deep networks, reducing overall query cost by more than two orders of magnitude compared to existing methods.

4.4 Generalization to ReLU-like Piecewise Linear Activation Functions

Our proposed Geometric Critical Point Screening method has layer-wise screening criteria that solely rely on the layer-wise geometric linearity of critical hyperplanes in the input space, and are completely decoupled from the numerical operation rules of specific activation functions. For all element-wise piecewise linear ReLU-like activation functions, including Leaky ReLU, PReLU [6], the critical point of a neuron is uniformly defined as the input point where the pre-activation value satisfies $V(\eta; x) = 0$. Therefore, the spatial position of the critical hyperplane is exclusively determined by the weight and bias parameters of the corresponding neuron, independent of the numerical properties of the activation function such as piecewise slope and saturation interval. When

the standard ReLU is replaced with any of the above ReLU-like activations, the position of all neurons' critical hyperplanes remains completely unchanged.

Leveraging this property, our method can quickly and accurately screen critical points belonging to the first and second hidden layer neurons without any modifications to the core algorithm, decision thresholds, or search pipelines. This fundamentally eliminates the redundant signature computation for massive non-target layer critical points in the conventional compute-first, cluster-later paradigm, ultimately achieving a substantial reduction in query cost and a significant improvement in extraction efficiency.

5 Experiments

All experiments are conducted on neural networks trained on the standard MNIST and CIFAR-10 datasets [11, 12].

Below we present experimental results of signature extraction on neural networks trained on MNIST or CIFAR-10 using the two methods described in Sections 4 and 5.

5.1 First Hidden Layer Signature Extraction

Table 1 shows the experimental results of first hidden layer signature extraction on different models. It can be seen that our method significantly outperforms Liu et al.'s method in both time and query cost.

Table 1 compares the performance of the proposed Geometric Critical Point Screening method and the end-to-end deep neural network extraction method by Liu et al. [13] for first hidden layer signature extraction on networks with different architectures. The analysis is conducted from two core dimensions: input layer dimension and first hidden layer dimension.

Impact of Input Layer Dimension. An increase in input layer dimension leads to a linear rise in the query overhead per critical point for signature extraction. Traditional methods require computing signatures for all candidate critical points obtained via global random search, which include a large number of non-target deep-layer neuron critical points. As a result, their query cost and execution time grow significantly with increasing input dimension. In contrast, the proposed method pre-screens critical points using geometric relationships and only computes signatures for the final identified target critical points of the first hidden layer, avoiding redundant queries on deep-layer non-target critical points. Comparing experimental results across different input dimensions reveals that the query cost savings of our method over traditional methods further expand as the input layer dimension increases.

Impact of First Hidden Layer Dimension. When the first hidden layer dimension increases, the query cost of traditional methods remains largely unchanged as they still need to perform signature extraction on a fixed number

of candidate critical points. However, the time cost of the clustering step increases significantly with dimension. For the proposed method, the time and query costs do increase moderately as more target critical points need to be processed for signature recovery, but the growth rate is gentle and controllable, far lower than the clustering time increase of traditional methods. Taking the MNIST $784-16^{(8)}-1$ network as an example, our method reduces the extraction time from 6 minutes 17 seconds to 25 seconds, and the query cost from $2^{21.60}$ to $2^{18.28}$, demonstrating a significant advantage.

In summary, the proposed method consistently outperforms traditional methods in both query cost and time efficiency, regardless of whether the input layer dimension or the first hidden layer dimension increases.

Table 1. First Hidden Layer Signature Extraction On Multiple Models

Model	Approach	Signature Extraction Time	Signature Query Cost
MNIST $784-8^{(8)}-1$	Liu et al. [13]	3m24s	$2^{21.59}$
	Ours	3.5s	$2^{16.66}$
MNIST $784-16^{(8)}-1$	Liu et al. [13]	6m17s	$2^{21.60}$
	Ours	25s	$2^{18.28}$
CIFAR-10 $3072-8^{(8)}-1$	Liu et al. [13]	56m48s	$2^{23.55}$
	Ours	1m24s	$2^{19.40}$

5.2 Second Hidden Layer Signature Extraction

To properly evaluate the effectiveness of our method and eliminate interference from floating-point errors in the first layer extraction stage, this experiment uses the true parameters of the first layer of the neural network.

Table 2 compares the performance of the proposed Geometric Critical Point Screening method and the end-to-end deep neural network extraction method by Liu et al. [13] for second hidden layer signature extraction on networks with different architectures.

Traditional methods rely on global random search to obtain critical points, which only correspond to partial activation states of the first hidden layer. Each such critical point can only extract partial signatures of neurons in the second hidden layer. As the dimension of the first hidden layer increases, the number of possible activation combinations of the first hidden layer grows exponentially. Traditional methods need to search for and merge a significantly increased number of critical points to extract complete signatures of the second hidden layer, which further amplifies their query overhead and computational cost in a superlinear manner.

In sharp contrast, the proposed method constructs a fully activated space of the first hidden layer, ensuring that all neurons in the first hidden layer are in the

activated state within the search space. A single critical point in this space can directly extract the complete signature of a second hidden layer neuron without the need for additional multi-critical-point signature merging operations.

Experimental results demonstrate that the proposed method significantly outperforms traditional methods in both query cost and time efficiency for second hidden layer signature extraction. It is precisely this core property—that a single critical point in the fully activated space can extract a complete signature—that enables our method to achieve a more remarkable performance improvement in the second hidden layer extraction stage than in the first hidden layer.

Table 2. Second Hidden Layer Signature Extraction On Multiple Models

Model	Approach	Signature Extraction Time	Signature Query Cost
MNIST $784 - 8^{(8)} - 1$	Liu et al. [13]	7m5s	$2^{24.21}$
	Ours	16s	$2^{14.48}$
MNIST $784 - 16^{(8)} - 1$	Liu et al. [13]	26m13s	$2^{24.22}$
	Ours	43s	$2^{17.02}$
CIFAR-10 $3072 - 8^{(8)} - 10$	Liu et al. [13]	1h33m	$2^{26.15}$
	Ours	23s	$2^{14.77}$
MNIST $784 - 32^{(8)} - 1$	Ours	90s	$2^{19.11}$

6 Conclusion

In this work, we made fundamental improvements to critical point localization and screening methods based on Carlini et al.'s cryptanalytic model extraction framework [4]. We proved that it is not necessary to compute signatures one by one; instead, we can effectively filter critical points of target layers solely using the geometric linear relationships of points in the input space. By eliminating meaningless signature extraction, we significantly improved the speed and reduced the query cost required for signature extraction, and achieved excellent results on deeper and wider neural networks.

Our method is applicable to the vast majority of practically deployed neural network architectures (satisfying input dimension greater than first hidden layer dimension). Although our method currently primarily works for extracting the first two layers of neural networks, it provides a very significant improvement to overall performance. As the depth and width of neural networks increase, our method will exhibit even better performance compared to random critical point search.Future work will focus on extending the geometric screening method to critical point screening for third and deeper-layers.

References

1. Canales-Martínez, I.A., Chávez-Saab, J., Hambitzer, A., Rodríguez-Henríquez, F., Satpute, N., Shamir, A.: Polynomial time cryptanalytic extraction of neural network models. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 3–33. Springer (2024)
2. Canales-Martínez, I.A., Santos, D.: Extracting some layers of deep neural networks in the hard-label setting. In: Escudero, D., Damgård, I. (eds.) Progress in Cryptology – LATINCRYPT 2025. pp. 399–421. Springer Nature Switzerland, Cham (2026)
3. Carlini, N., Chávez-Saab, J., Hambitzer, A., Rodríguez-Henríquez, F., Shamir, A.: Polynomial time cryptanalytic extraction of deep neural networks in the hard-label setting. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 364–396. Springer (2025)
4. Carlini, N., Jagielski, M., Mironov, I.: Cryptanalytic extraction of neural network models. In: Annual international cryptology conference. pp. 189–218. Springer (2020)
5. Chen, Y., Dong, X., Guo, J., Shen, Y., Wang, A., Wang, X.: Hard-label cryptanalytic extraction of neural network models. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 207–236. Springer (2024)
6. Chen, Y., Dong, X., Ma, R., Shen, Y., Wang, A., Yu, H., Wang, X.: Delving into cryptanalytic extraction of prelu neural networks. In: Hanaoka, G., Yang, B.Y. (eds.) Advances in Cryptology – ASIACRYPT 2025. pp. 576–607. Springer Nature Singapore, Singapore (2026)
7. Daniely, A., Granot, E.: An exact poly-time membership-queries algorithm for extracting a three-layer reLU network. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=-CoNloheTs
8. Fefferman, C.: Reconstructing a neural net from its output. Revista Matemática Iberoamericana **10**(3), 507–555 (1994). https://doi.org/10.4171/RMI/160
9. Foerster, H., Mullins, R., Shumailov, I., Hayes, J.: Beyond slow signs in high-fidelity model extraction. Advances in Neural Information Processing Systems **37**, 19496–19522 (2024)
10. Hugging Face Team: Hugging face. https://huggingface.co/ (nd), accessed: 2025-02-12
11. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 and cifar-100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html (2009)
12. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Handwritten digit recognition with a back-propagation network. In: Touretzky, D. (ed.) Advances in Neural Information Processing Systems. vol. 2. Morgan-Kaufmann (1989)
13. Liu, H., Siproudhis, A., Experton, S., Lorenz, P., Boura, C., Peyrin, T.: Navigating the deep: End-to-end extraction on deep neural networks. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 482–512. Springer (2026)
14. Lowd, D., Meek, C.: Adversarial learning. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. pp. 641–647 (2005)

15. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning (2010), https://api.semanticscholar.org/CorpusID:15539264
16. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security. pp. 506–519 (2017)
17. Reith, R.N., Schneider, T., Tkachenko, O.: Efficiently stealing your machine learning models. In: Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society. pp. 198–210 (2019)
18. Rolnick, D., Kording, K.: Reverse-engineering deep relu networks. In: International conference on machine learning. pp. 8178–8187. PMLR (2020)
19. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction {APIs}. In: 25th USENIX security symposium (USENIX Security 16). pp. 601–618 (2016)
20. Xiao, T., Zhu, J.: Foundations of large language models. https://api.semanticscholar.org/CorpusID:275570622 (2025)