

End-to-End Hard-Label Cryptanalytic Model Extraction Using Efficient Sign Recovery

Akira Ito^[0000-0002-4602-7570], Takayuki Miura^[0000-0001-8694-312X], and
Yosuke Todo^[0000-0002-6839-4777]

¹ Tohoku University
2-1-1 Katahira, Aoba-ku, Sendai-shi, 980-8577, Japan
`akira.ito.b1@tohoku.ac.jp`,
² NTT Social Informatics Laboratories,
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan
`tkyk.miura@ntt.com`, `yosuke.todo@ntt.com`

Abstract. The importance of deep neural networks (DNNs) is widely recognized, and the parameters obtained through training are regarded as valuable assets. Recently, attacks that extract these parameters using only oracle queries to a DNN have been actively studied at IACR conferences. The hard-label setting is the most challenging setting for model extraction, where an adversary can observe only the final output label, such as “dog” or “cat”. At Eurocrypt 2025, Carlini et al. proposed polynomial-time hard-label extraction of ReLU-based MLPs. However, one step of this attack process, i.e., sign recovery, requires a large number of queries and substantial computation. Implementing this step in a black-box setting remains difficult. Consequently, a fully black-box end-to-end demonstration on trained deep ReLU MLPs has remained a challenge. In this paper, we propose a new sign-recovery algorithm based on a completely different principle from the existing method. Our method requires no dedicated queries for sign recovery. In our experiments, it achieves higher sign-recovery accuracy than the existing method. Consequently, it enables efficient sign recovery even for trained models. With our sign-recovery algorithm, all steps of hard-label model extraction can be implemented in a black-box setting. By combining these implementations, we demonstrate end-to-end model extraction from models trained on MNIST and Fashion-MNIST, with width 16 and 4 or 6 hidden layers, achieving over 98% label agreement.

Keywords: ReLU network · Model extraction · Hard-label attack · Sign recovery · End-to-End.

1 Introduction

Deep neural networks (DNNs) have brought substantial advances across domains such as image processing, audio signal processing, and natural language processing [19,15,16]. Nowadays, AI-based services have become an important part of modern digital infrastructure. However, training these models requires

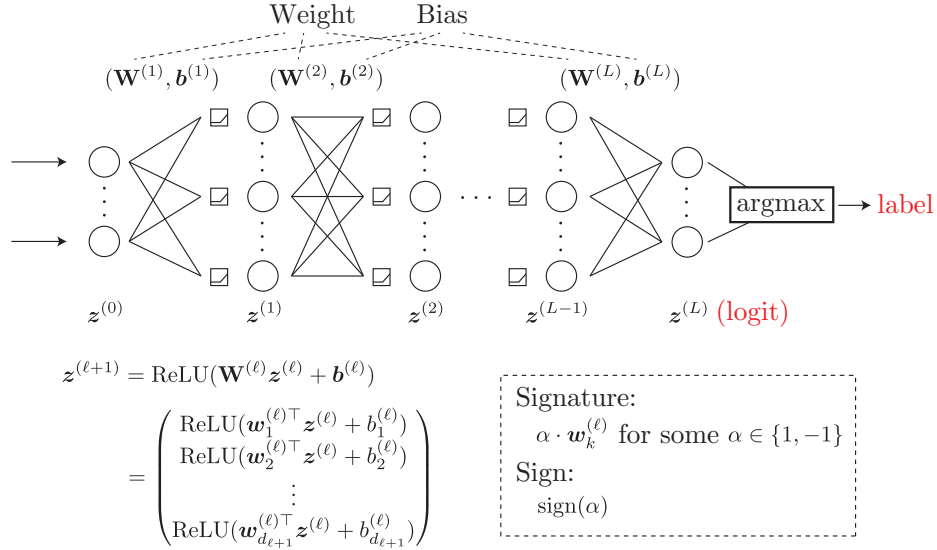


Fig. 1: ReLU-based MLP and model extraction.

substantial computational resources and large datasets, so the learned parameters are typically treated as confidential intellectual property. If an attacker extracts the parameters of a trained model, it poses a significant risk to the model provider [11,9,6,5,13,18,17,14].

Recently, cryptanalytic approaches to model extraction have attracted attention at IACR venues, motivated by structural similarities between multilayer perceptrons (MLPs) and symmetric-key ciphers [3,12,4,2,10,8]. Both can be viewed as compositions of linear and nonlinear transformations; the hidden information is a cryptographic key in one case and neural network parameters in the other. From this perspective, cryptanalytic techniques that adaptively query an oracle to recover secret information can be naturally applied to model extraction.

This line of research was initiated by Carlini et al. [3], who focused on ReLU-based MLPs because of their widespread use. Fig. 1 shows the ReLU-based MLP. In a canonical representation of a ReLU-based MLP, the adversary’s goal is to extract the *signature*, *sign*, and *bias* of each target neuron layer by layer. The original CRYPTO 2020 work assumed that the adversary can observe raw outputs (logits), and this setting is called the *soft-label setting*. In the soft-label setting, a polynomial-time algorithm was proposed at Eurocrypt 2024 [12]. The feasibility of end-to-end extraction was subsequently demonstrated on trained models at Eurocrypt 2026 [10].

The hard-label setting is a different and more restrictive setting. Logits, which are visible in the soft-label setting, are usually not exposed to users, and only the output label is visible. In the hard-label setting, the adversary attempts to recover internal parameters using only these labels. Thus, the hard-label setting

reveals strictly less information than the soft-label setting. The initial work on the hard-label setting appeared at Asiacrypt 2024, where sign recovery required exponential cost [4]. Later, at Eurocrypt 2025, Carlini et al. proposed algorithms with polynomial running time [2]. They first collect many *intersection spaces*. Each intersection space is formed by the intersection of an activation boundary, where a neuron switches between active and inactive, and a decision boundary, which separates two output labels. Each intersection space is represented by an intersection point, together with the normal vectors of its two adjacent decision boundaries. Then, the intersection spaces associated with the same neuron are grouped. From each group, the target neuron’s signature and corresponding bias are recovered up to a common sign. Finally, this remaining sign ambiguity is resolved using the method that we call the *boundary-walking method*.

Difficulty of Sign Recovery in the Hard-Label Setting. Fig. 4 illustrates the boundary-walking method. Starting from an intersection point $\mathbf{s}$ for the target neuron, the method pulls the recovered target signature $\alpha\mathbf{w}_t$ and its opposite $-\alpha\mathbf{w}_t$ back to input-space directions through the transpose $\mathbf{\Gamma}^\top$ of the local linear map. It projects each direction onto the corresponding local decision boundary and walks along the boundary. When it recognizes a bend in the decision boundary, it determines whether the bend is caused by a neuron-state change in a previous layer, called a *past toggle*, or a subsequent layer, called a *future toggle*. If it is a past toggle, it recovers the new decision-boundary normal and resumes walking. The walk continues until the future toggle is detected. The total distance from the intersection point to the future toggle is measured. The method relies on the heuristic observation that the on-side direction, i.e., $\mathbf{w}_t$, tends to affect subsequent layers more strongly and hence reaches a future toggle after a shorter distance. According to [2], sign recovery aggregates the results from 10^2 – 10^3 intersection points.

A black-box implementation of the boundary-walking method is challenging³. We assume that intersection points and their two adjacent decision-boundary normals have already been recovered with sufficient precision, since signature recovery requires the same information. Then, it might be possible to walk along a decision boundary and recognize its first bends. However, it must distinguish a bend caused by a neuron toggle from a false bend caused by estimation noise. It also must avoid crossing multiple toggles at once. Thus, reliable bend detection requires a substantial number of queries. After a past toggle, the new decision-boundary normal must be recovered again in the black-box setting. Then, the original decision boundary may remain close to the point immediately after the toggle, which makes accurate recovery of the new normal more difficult. This precision cannot be compromised, because otherwise the procedure cannot reliably resume walking along the boundary. Consequently, the boundary-walking

³ Recently, the ePrint version of [2] was updated to include a black-box end-to-end hard-label model-extraction demonstration [1]. However, this result should be interpreted carefully because their target is an artificially constructed model whose weights are not learned from data. For details, refer to Section A.

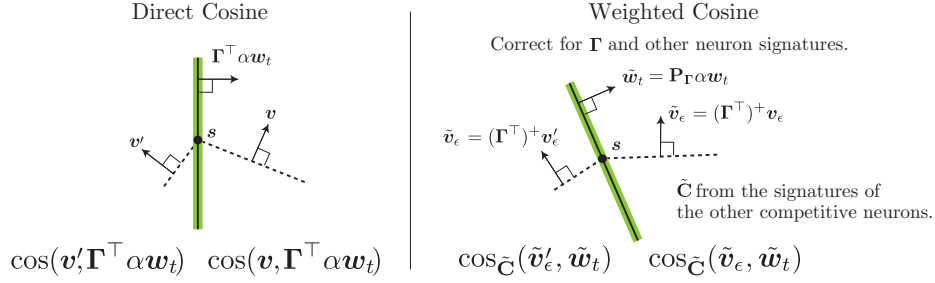


Fig. 2: Overview of the new sign-recovery method.

method repeatedly invokes query-intensive and computationally expensive sub-routines. For their trained-model experiments, Carlini et al. [2] evaluated this procedure through a proof-of-concept white-box implementation. Then, decision-boundary normals are computed analytically and toggles are detected from internal states without error. Even their white-box experiments aggregate results from 10^2 – 10^3 intersection points to recover the sign. In a fully black-box setting, errors in estimating normals and detecting bends can require more samples.

1.1 Our Contribution

New Sign-Recovery Method: Cosine Method. Our main contribution is a new sign-recovery algorithm based on a completely different principle from the boundary-walking method. Our proposed method, called the *cosine method*, requires neither additional queries nor past/future toggle detection. It only performs offline computations on the given intersection spaces. In our experiments, aggregating 50 intersection spaces recovers most signs correctly, whereas the boundary-walking method requires substantially more samples. Since a certain number of intersection spaces have already been collected for signature recovery, in many cases, it requires no additional queries. It only reuses the collected intersection spaces and performs offline computations.

Fig. 2 shows the quantity measured by the cosine method. Concretely, the cosine method compares the cosine similarities between the target signature pulled back to the input space, $\Gamma^\top \alpha w_t$, and the two decision-boundary normals v and v' associated with an intersection space. We call this simple case the *direct cosine method*. The sign is recovered from the observation that $\cos(\Gamma^\top w_t, v_{\text{on}})$ has a larger variance than $\cos(\Gamma^\top w_t, v_{\text{off}})$. As is clear from this procedure, the proposed method uses no additional queries given intersection spaces.

Although comparing cosine similarities is simple, the direct cosine method quickly becomes ineffective in deeper layers. This is because the local linear map Γ is typically highly distorted, and the cosine similarities become dominated by the anisotropy of Γ . Moreover, in trained models, neuron signatures are anisotropically distributed, and decision-boundary normals themselves can be biased toward particular directions. We therefore introduce a more advanced

weighted cosine method. The weighted cosine method projects the quantities into the image space of the local linear map and corrects them using a matrix determined by the non-target signatures in the same layer. Although it requires more computation than the direct cosine method, it still requires no queries, and its computational cost is negligible compared with intersection-point collection. Consequently, the proposed method is substantially faster than the existing boundary-walking method, even when compared with its white-box implementation.

How many intersection spaces are required to recover a sign using the cosine method? To answer this question, we estimate the variance of the cosine statistics. The resulting predictions are experimentally verified on both random models and trained MNIST models. Our analysis predicts a useful statistical separation at approximately 50 intersection spaces under the stated approximations. On an MNIST-trained model with five hidden layers of width 128, our method correctly recovers 506 of 510 signs in layers 2–5, excluding persistent and dead neurons, using 50 intersection spaces per neuron (Table 2). Thus, compared with the boundary-walking method, the proposed method requires no additional queries, has low computational cost, and achieves high accuracy.

End-to-End Hard-Label Model Extraction. Our second contribution is, to the best of our knowledge, the first demonstration of fully black-box end-to-end hard-label model extraction on trained deep ReLU MLPs, enabled by incorporating the sign-recovery algorithm described above. The feasibility of end-to-end model extraction has already been demonstrated in the soft-label setting [10]. In the hard-label setting, however, end-to-end extraction of trained deep ReLU MLPs has remained a challenge, despite the demonstration on a specially structured artificial model discussed in Section A.

The obstacles extend beyond sign recovery to signature recovery. In deeper layers, biased activation patterns can cause intersection spaces associated with different neurons to pass the consistency check, producing spurious signatures. Moreover, the collected intersection spaces may not determine every coordinate, leaving some signatures only partially recovered. Similar issues have been reported in the soft-label setting [10], but addressing them is more difficult when only output labels are observable. The hard-label setting also introduces a distinct source of false positives: multiple intersection spaces can share a decision-boundary hyperplane in the input space of an intermediate layer, causing its normal to be mistaken for a signature.

We analyze these issues and combine methods that address them into an end-to-end attack. For sign recovery, we use our proposed sign-recovery methods and correct erroneous assignments by checking their consistency with the decision-boundary normals already collected. For signature recovery, we identify the main cases in which consistency checks produce false positives and reject spurious candidates by examining bends in the decision boundary at candidate activation boundaries and the relationship between candidate signatures and decision-boundary normals. To recover missing coordinates, we actively search

Table 1: Notation

Symbol	Description
$f : \mathbb{R}^{d_0} \rightarrow \mathbb{R}^{d_L}$	ReLU network with L layers
$\sigma : \mathbb{R}^{d_*} \rightarrow \mathbb{R}^{d_*}$	ReLU function applied component-wise
$\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$	weight matrix in the ℓ th layer
$\mathbf{b}^{(\ell)} \in \mathbb{R}^{d_\ell}$	bias vector in the ℓ th layer
$\mathbf{h}^{(\ell)}(\mathbf{x}) \in \mathbb{R}^{d_\ell}$	output vector for input $\mathbf{x}$ after the ℓ th layer
$\mathbf{\Gamma}_{\mathbf{x}}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_0}$	forward local linear matrix up to the ℓ th layer for input $\mathbf{x}$
$\mathbf{s} \in \mathbb{R}^{d_0}, \quad \mathcal{S} \subset \mathbb{R}^{d_0}$	intersection point, intersection space

for informative intersection spaces and apply existing cross-layer extraction techniques [8] to obtain the missing weight information from activation boundaries in the next layer.

Together, these analyses and methods enable end-to-end extraction from models of width 16 with 4 and 6 hidden layers trained on MNIST and FMNIST. Our attack pipeline recovers all neurons except dead and almost-dead neurons and achieves over 98% output agreement with the victim model for every evaluated model on 100,000 inputs drawn from a standard Gaussian distribution.

2 Preliminaries

The notation used throughout this paper is summarized in Table 1.

2.1 ReLU Network and Local Linearity

Definition 1 (ReLU [7]). *The ReLU function $\sigma : \mathbb{R}^{d_*} \rightarrow \mathbb{R}^{d_*}$ is defined component-wise by $\sigma(\mathbf{x})_i = \max\{\mathbf{x}_i, 0\}$ for each $1 \leq i \leq d_*$.⁴*

Definition 2 (ReLU Network). *Let $L \geq 2$ be the number of layers, including the affine output layer. For $1 \leq \ell \leq L - 1$, define*

$$\mathbf{h}^{(\ell)}(\mathbf{x}) = \sigma(\mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)}(\mathbf{x}) + \mathbf{b}^{(\ell)}),$$

where $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$, $\mathbf{b}^{(\ell)} \in \mathbb{R}^{d_\ell}$, and $\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x}$. The input and output of the ReLU function are referred to as the pre-activation and activation, respectively. Thus, the pre-activation at layer ℓ is $\mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)}(\mathbf{x}) + \mathbf{b}^{(\ell)}$, and the corresponding activation is $\mathbf{h}^{(\ell)}(\mathbf{x})$. The output layer is defined by $\mathbf{h}^{(L)}(\mathbf{x}) = \mathbf{W}^{(L)}\mathbf{h}^{(L-1)}(\mathbf{x}) + \mathbf{b}^{(L)}$, where $\mathbf{W}^{(L)} \in \mathbb{R}^{d_L \times d_{L-1}}$ and $\mathbf{b}^{(L)} \in \mathbb{R}^{d_L}$. An L -layer ReLU network is the function $f : \mathbb{R}^{d_0} \rightarrow \mathbb{R}^{d_L}$ given by $f(\mathbf{x}) = \mathbf{h}^{(L)}(\mathbf{x})$. The parameter set is $\theta = \{(\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)})\}_{\ell=1, \dots, L}$. The weight vector of the k th neuron in layer ℓ , denoted by $\mathbf{w}_k^{(\ell)} \in \mathbb{R}^{d_{\ell-1}}$, is the transpose of the k th row of $\mathbf{W}^{(\ell)}$. For a hard-label model,

⁴ Strictly speaking, the definition depends on the input dimension; however, it is standard to use the same symbol σ to denote such functions, as they apply the same operation to each component of the input vector.

let $\hat{y} : \mathbb{R}^{d_L} \rightarrow [d_L]$ be the function selecting a largest-logit class, with a fixed rule for breaking ties. The resulting classifier is $\hat{y} \circ f : \mathbb{R}^{d_0} \rightarrow [d_L]$.

A ReLU network partitions the input space into regions according to the activation/inactivation patterns of its neurons, and within each region, the network represents an affine transformation. Such a region is called a *ReLU cell*. By the positive homogeneity of ReLU, positive neuron-wise rescalings can be propagated through the network without changing the function it represents [2]. Hence, in the hard-label setting, we assume without loss of generality that each nonzero row vector of a hidden-layer weight matrix has unit norm, with its bias rescaled by the same factor and the corresponding column of the next-layer matrix adjusted inversely. The final-layer weight matrix is left unnormalized. We refer to this normalized form as the *canonical representation*.

Since a ReLU network is locally an affine transformation, it exhibits locally linear behavior. In particular, we define matrices that characterize the local linearity up to each intermediate layer.

Definition 3 (Forward Local Linear Matrix). Let $f : \mathbb{R}^{d_0} \rightarrow \mathbb{R}^{d_L}$ be a ReLU network defined as above. Let $\mathbf{x} \in \mathbb{R}^{d_0}$, and $1 \leq \ell \leq L$. Then, there exist $\mathbf{\Gamma}_{\mathbf{x}}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_0}$ and $\mathbf{b}_{\mathbf{x}}^{(\ell)} \in \mathbb{R}^{d_\ell}$ such that $\mathbf{h}^{(\ell)}(\mathbf{y}) = \mathbf{\Gamma}_{\mathbf{x}}^{(\ell)} \mathbf{y} + \mathbf{b}_{\mathbf{x}}^{(\ell)}$ for any $\mathbf{y}$ in the same ReLU cell as $\mathbf{x}$. The matrix $\mathbf{\Gamma}_{\mathbf{x}}^{(\ell)}$ is called the forward local linear matrix at $\mathbf{x}$. Set $\mathbf{\Gamma}_{\mathbf{x}}^{(0)} = \mathbf{I}_{d_0}$ for all $\mathbf{x} \in \mathbb{R}^{d_0}$, where $\mathbf{I}_{d_0}$ denotes the $d_0 \times d_0$ identity matrix.

2.2 Cryptanalytic Model Extraction

The goal of cryptanalytic model extraction is to recover the parameters of a target model as precisely as possible from oracle access to its predictions, up to transformations that preserve its observable behavior.

Adversarial Capabilities and Objectives. In line with [2], we consider the following attacker capabilities and goals. The attacker queries an oracle $\mathcal{O}(\mathbf{x}) = \hat{y}(f_\theta(\mathbf{x}))$ on adaptively selected inputs and observes only the corresponding class labels. The goal is to reconstruct parameters $\hat{\theta}$ that reproduce the target classifier as accurately as possible. Hard-label access does not uniquely determine logits: adding a common affine function to all logits or multiplying them by a common positive scalar preserves every label. In particular, functional model extraction is defined in terms of functional equivalence.

Definition 4 (Hard-label functional equivalence). Let f_θ and $\hat{f}_{\hat{\theta}}$ be the target and extracted logit models, and let $\mathcal{D}_X$ be a probability distribution supported on $X \subseteq \mathbb{R}^{d_0}$. The corresponding classifiers are ϵ -functionally equivalent with respect to $\mathcal{D}_X$ if $\Pr_{\mathbf{x} \sim \mathcal{D}_X} [\hat{y}(f_\theta(\mathbf{x})) = \hat{y}(\hat{f}_{\hat{\theta}}(\mathbf{x}))] \geq 1 - \epsilon$.

Accordingly, we evaluate functional extraction by label agreement under a specified input distribution, separately from parameter-recovery accuracy.

Furthermore, consistent with prior work, we adopt the following standard assumptions. **Architecture knowledge:** The attacker is aware of the model

architecture, including the number of layers and units. **Unrestricted inputs:** The attacker can query arbitrary inputs from $\mathbb{R}^{d_0}$. **Precise computation:** The DNN is executed with sufficiently high-precision floating-point arithmetic. **Fully connected ReLU network:** The target is a multilayer perceptron with fully connected layers, ReLU hidden activations, and an affine output layer.

2.3 Overview of Hard-Label Model Extraction

The hard-label model extraction attack proposed by Carlini et al. [2] explores the decision boundary and leverages *intersection spaces*, where the decision boundary intersects activation boundaries—i.e., boundaries at which neuron activations switch—to reconstruct neuron parameters.

Definition 5 (Intersection space and intersection point). *For a ReLU network, the intersection between an activation boundary and a decision boundary is termed an intersection space. A point lying within an intersection space is referred to as an intersection point.*

Let $\mathcal{S} \subset \mathbb{R}^{d_0}$ be this affine space. It extends the local intersection beyond the adjacent ReLU cells; its distant points need not lie on the network’s actual boundaries. We use two equivalent representations: $(\mathbf{s}, \mathbf{v}, \mathbf{v}')$, where $\mathbf{s}$ is an intersection point and $\mathbf{v}, \mathbf{v}'$ are the two adjacent normals, and $(\mathbf{s}, \mathbf{N})$, where the columns of $\mathbf{N} \in \mathbb{R}^{d_0 \times (d_0-2)}$ form an orthonormal basis of $\text{span}\{\mathbf{v}, \mathbf{v}'\}^\perp$.

Details on how to collect intersection spaces are provided in Section B.

Overview of hard-label model extraction. An overview of the attack is as follows.

1. Collect a large number of intersection spaces.
2. Starting from the shallowest layer, repeat the following steps:
 - Identify pairs of intersection spaces that correspond to the same activation boundary (Section 2.4).
 - Group intersection spaces corresponding to the same activation boundary and recover the signature of the neuron associated with that boundary (Section 2.4).
 - Recover the signs of the collected signatures (Section 2.5).
 - For signatures that cannot be fully recovered, defer their identification to the cross-layer extraction step [8].
3. Proceed layer by layer from shallow to deep layers.

2.4 Signature Recovery

In the input space of its layer, a neuron’s weight vector is normal to its activation-boundary hyperplane. Thus, given multiple intersection spaces on the same activation boundary, as illustrated in Fig. 3, we can reconstruct the boundary and recover a nonzero scalar multiple of the weight vector, which we call its *signature*.

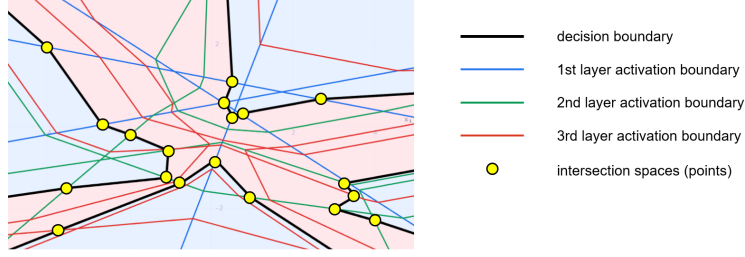


Fig. 3: Illustration of a two-dimensional input space.

Definition 6 (Signature). Let $\mathbf{w}_k^{(\ell)} \in \mathbb{R}^{d_{\ell-1}}$ be the weight vector of the k th neuron in layer ℓ , i.e., the transpose of the k th row of $\mathbf{W}^{(\ell)}$. Its signature is $\alpha \mathbf{w}_k^{(\ell)}$ for $\alpha \in \{1, -1\}$.⁵

Signature recovery consists of two stages. First, a consistency check tests whether two mapped intersection spaces satisfy a common affine constraint; the procedure is given in `ISCONSISTENT()` (Algorithm 2) in Section C. Then, after collecting sufficient consistent intersection spaces, the corresponding signature is reconstructed using `RECOVERSIGNATURE()` (Algorithm 3) in Section C.

The consistency check is performed as follows. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be intersection spaces with basis matrices $\mathbf{N}_1, \mathbf{N}_2$ and representative points $\mathbf{s}_1, \mathbf{s}_2$. For $i \in \{1, 2\}$, let $\mathbf{T}_i = \mathbf{\Gamma}_{\mathbf{s}_i}^{(\ell-1)} \mathbf{N}_i$ and $\mathbf{u}_i = \mathbf{h}^{(\ell-1)}(\mathbf{s}_i)$ be their mapped directions and representative points in the input space of layer ℓ . Let ν be the number of coordinates corresponding to neurons in layer $\ell - 1$ that are active at either point; for $\ell = 1$, set $\nu = d_0$. The test accepts if $\text{rank}[\mathbf{T}_1, \mathbf{T}_2, \mathbf{u}_1 - \mathbf{u}_2] < \nu$. This detects a common affine constraint on the active coordinates. It is necessary for a shared activation hyperplane whose normal has a nonzero restriction to these coordinates, but it is not sufficient to identify a common target neuron. In particular, the spurious groups analyzed in Section 4 can also pass this test.

Now suppose that $\mathcal{S}_1, \dots, \mathcal{S}_m$ correspond to the same target neuron. With $\mathbf{T}_i$ and $\mathbf{u}_i$ defined as above, form

$$\mathbf{U} = [\mathbf{T}_1, \dots, \mathbf{T}_m, \mathbf{u}_2 - \mathbf{u}_1, \dots, \mathbf{u}_m - \mathbf{u}_1].$$

If $\text{rank}(\mathbf{U}) = d_{\ell-1} - 1$, the one-dimensional null space of $\mathbf{U}^\top$ determines the signature up to sign. Let $\mathbf{w}$ be a unit vector in this null space. The corresponding bias is $b = -\mathbf{w}^\top \mathbf{u}_1$. The representative-point differences are needed to enforce a common affine hyperplane, in addition to orthogonality to the mapped directions. If the rank is smaller, the signature is not uniquely determined; additional informative spaces or the methods in Section 4 are needed. A larger rank indicates inconsistent data in exact arithmetic. All ranks and null spaces are computed using singular value decomposition (SVD).

⁵ As discussed earlier, the norm can be normalized without loss of generality, and hence it suffices to consider the values 1 and -1 .

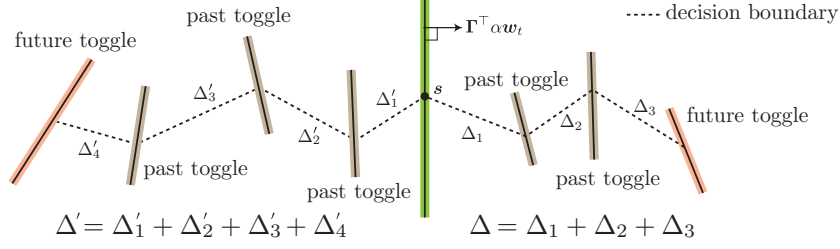


Fig. 4: Existing sign-recovery method. Compare the distance between Δ and Δ' .

2.5 Sign Recovery

Signature recovery determines a signature $\alpha \cdot \mathbf{w}_k^{(\ell)}$ for the k th neuron in the ℓ th layer, and the remaining unknown variable is the sign, $\alpha \in \{1, -1\}$. Recovering this sign is necessary to determine which side of the corresponding activation boundary is the active side of the ReLU.

The existing sign-recovery method is based on a heuristic observation: when a carefully chosen direction changes the target neuron’s pre-activation by ϵ from a point on an activation boundary, this perturbation ϵ affects the subsequent layer if the direction is on the active (on) side, whereas it is zeroed out by ReLU if it is on the inactive (off) side. In the soft-label setting [12], the sign is successfully recovered by measuring the magnitude of the difference between the output logits. Although the same concept is applied in the hard-label setting [2], it measures a more indirect distance because logits are unobservable.

We call the existing method the *boundary-walking method*, and Fig. 4 summarizes the behavior.

1. Starting from an intersection point $\mathbf{s}$, pull the target signature back to the input space⁶ as $(\mathbf{\Gamma}_s^{(\ell-1)})^\top \alpha \mathbf{w}_t^{(\ell)}$. Project this direction onto the corresponding decision boundary.
2. Walk along the decision boundary until a bend is detected. If it is caused by a past toggle, i.e., a neuron state switches before the target layer, recover the new decision-boundary normal after the bend and resume walking.
3. If a bend in the decision boundary is identified and its cause cannot be explained by past toggles, it supposes future toggles, i.e., a neuron state switches after the target layer. The total distance of observing the future toggle is measured.
4. Repeat in the opposite direction and compare the two distances.

In other words, the distance to a future toggle is used instead of the magnitude of the logit difference. Considering the underlying heuristic observation, it is expected that the on side will toggle a future neuron’s state sooner.

⁶ An alternative is *perfect control*. A direction $\Delta \mathbf{x}$ satisfying $\mathbf{\Gamma} \Delta \mathbf{x} = \alpha \mathbf{e}_t$ changes only the target pre-activation. Such a direction exists when $\mathbf{e}_t \in \text{Im}(\mathbf{\Gamma})$; full row rank is sufficient. However, the condition is rarely satisfied in deeper layers.

Note that a faster future toggle on the on side does not necessarily hold in all intersection spaces, since these movements also change other neurons’ outputs in the target layer and they also affect subsequent layers. The procedure is therefore repeated over multiple intersection points corresponding to the same target neuron. Two distances are compared for every intersection point, and a direction with a shorter distance gets one vote. If repeated past toggles prevent a reliable comparison, the corresponding intersection point is discarded. The authors of [2] estimated the number of intersection points required for high-confidence sign recovery through (white-box) experiments. While 100 points are sufficient for the first layer, approximately 1,000 points are required for the second and subsequent layers.

3 Efficient Sign Recovery without Dedicated Queries

3.1 Motivation

Hard-label model extraction by Carlini et al. [2] is conceptually implementable, but its black-box implementation remains challenging. In particular, the implementation of the boundary-walking method causes significant difficulty.

First, a decision-boundary normal recoverable in a black-box setting inevitably contains some noise. The boundary-walking method is highly sensitive to the noise of decision-boundary normals. When the normal is noisy, even moving along a decision boundary is not straightforward. One must distinguish a bend caused by a neuron toggle from a false bend caused by noise. The step size during movement is also important. An adversary does not know how far it must move before a toggle occurs. Large steps introduce a risk that some neuron toggles are overlooked. Small steps cause the number of queries required before a toggle to explode. We need to recover the new decision-boundary normal once we recognize a past toggle. A point immediately after a toggle is also close to the original decision boundary, so additional care is needed to obtain an accurate decision-boundary normal. Even after all these issues can be resolved efficiently and accurately, the same procedure must be performed on both the on and off sides until the future toggle and be repeated 10^2 – 10^3 times for each target neuron. Note that the number of iterations is estimated in a noise-free white-box implementation. We might need more points when noise is present.

In practice, the authors of [2] themselves provide only a white-box proof of concept for the trained model. The recent end-to-end implementation [1] is demonstrated on a specially constructed artificial model that facilitates sign recovery. These results leave the practical cost and robustness of fully black-box boundary walking on trained deep networks unresolved.

3.2 Intuition Behind the Direct Cosine Method

Our new sign-recovery algorithm is based on a completely different principle from existing methods. Unlike the existing method, it does not require dedicated queries. The method requires only intersection spaces, which are already

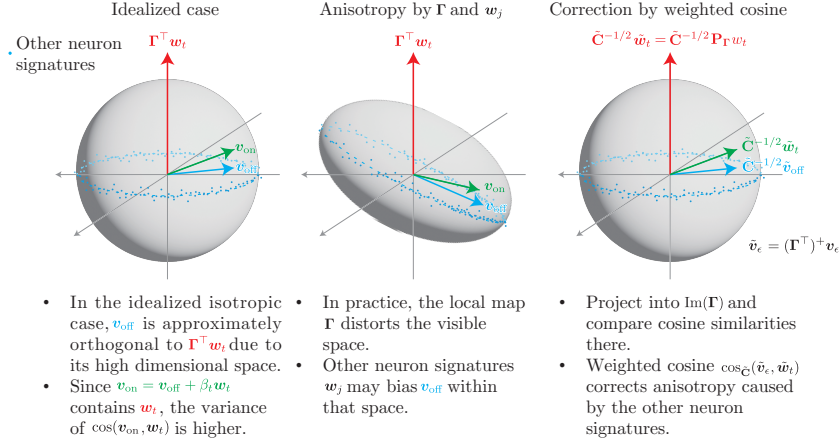


Fig. 5: Schematic three-dimensional visualization of the (weighted) cosine methods, obtained by taking the target-signature direction as the z -axis.

known to an adversary for signature recovery. Once they are given, just offline computation is sufficient.

The basic principle is as follows: We compute the cosine similarities between each decision-boundary normal and the recovered signature. Our method exploits the heuristic that the cosine similarity with the on-side decision-boundary normal has larger variance than that with the off-side normal.

Why does the on-side cosine similarity have a higher variance? Consider recovering the sign of the t th neuron in hidden layer ℓ . Let $w_t \in \mathbb{R}^{d_\ell-1}$ be the target weight vector, and let $w_j \in \mathbb{R}^{d_\ell-1}$ be the j th weight vector in the same layer. At an intersection point s , write $\Gamma = \Gamma_s^{(\ell-1)} \in \mathbb{R}^{d_{\ell-1} \times d_0}$. Then, using the index set $\mathcal{J}_{\text{act}}$ of neurons active in the same layer, the decision-boundary normal on the off side can be written as

$$v_{\text{off}} := \Gamma^\top \sum_{j \in \mathcal{J}_{\text{act}}} \beta_j w_j \in \mathbb{R}^{d_0}.$$

On the on side, we consider the same decision boundary with only the target neuron state switched, and thus its normal is

$$v_{\text{on}} := v_{\text{off}} + \beta_t \Gamma^\top w_t.$$

Thus, v_{on} contains the $\Gamma^\top w_t$ component.

A direct cosine is defined by $\cos(\Gamma^\top \alpha_t w_t, v_\epsilon)$, where $\epsilon \in \{\text{off}, \text{on}\}$. Both the target signature $\Gamma^\top \alpha_t w_t$ and the decision normal v_ϵ lie in $\text{Im}(\Gamma^\top) \subseteq \mathbb{R}^{d_0}$. The dimension of this subspace is $r_\Gamma := \dim \text{Im}(\Gamma^\top) = \text{rank}(\Gamma)$.

Assuming that $\Gamma^\top \alpha_t w_t$ and v_{off} are independent and isotropic in this subspace, as a typical behavior in high-dimensional space, we have

$$\cos(\Gamma^\top \alpha_t w_t, v_{\text{off}}) \stackrel{\text{approx}}{\sim} \mathcal{N}\left(0, \frac{1}{r_\Gamma}\right).$$

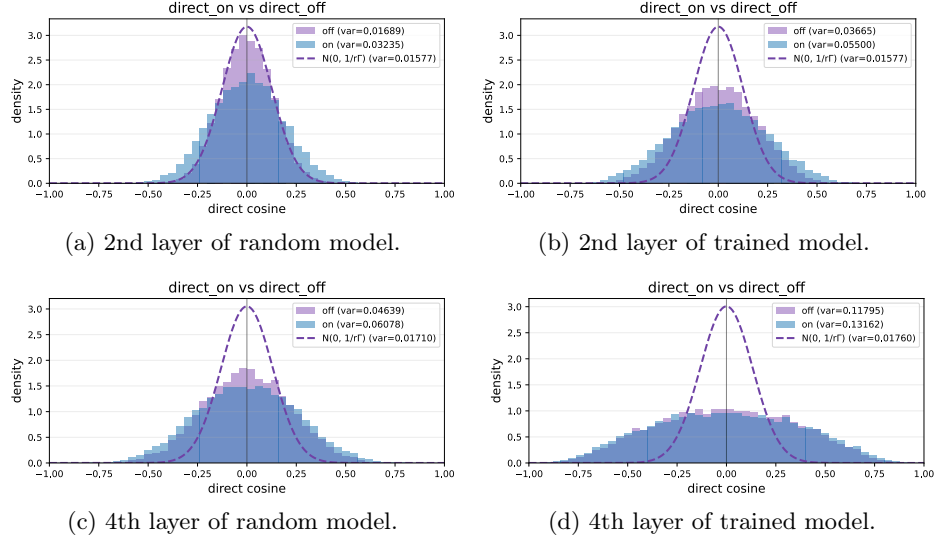


Fig. 6: Direct cosine distributions for random and MNIST-trained models.

The left figure of Fig. 5 illustrates this idealized geometry. Under the isotropic model, the target direction is nearly orthogonal to the active non-target directions whose linear combination forms the off-side normal.

The on-side normal additionally contains the target component $\beta_t \mathbf{\Gamma}^\top \mathbf{w}_t$. Thus, its cosine is expected to have larger variance than that of the off-side.

Experiments. We experimentally observed the direct cosine distributions for both initialized and MNIST-trained models with 784 inputs, 5 hidden layers of width 128, and 10 outputs. Fig. 6 summarizes the results, where 100 intersection spaces are used for each neuron, giving 12,800 spaces per evaluated layer.

The experiment for the 2nd layer of the initialized model shows a good fit to the random cosine assumption, as shown in Fig. 6(a). However, Fig. 6(c) shows that the fit rapidly degrades as the target layer becomes deeper. Moreover, it is not useful for the trained model (see Figs. 6(b) and 6(d)).

3.3 $\mathbf{\Gamma}$ -Weighted Cosine Method

Cosine similarity in the input space is strongly affected by the distortion of the local linear map $\mathbf{\Gamma}$, resulting in the direct cosine method not working in the deep layer. To mitigate this effect, we map the recovered decision-boundary normal $\mathbf{v}_\epsilon$ to the input space of the target layer.

From the input space, the target layer can be reached only through the local linear map $\mathbf{\Gamma}$. For example, when $r_\mathbf{\Gamma}$ is small, the target-layer normal cannot be reconstructed exactly. Therefore, comparison must be performed in the image of

$\mathbf{\Gamma}$, namely, in the space visible through the local linear map $\mathbf{\Gamma}$. The projection matrix associated with $\mathbf{\Gamma}$ is

$$P_{\mathbf{\Gamma}} := \mathbf{\Gamma}(\mathbf{\Gamma}^\top \mathbf{\Gamma})^+ \mathbf{\Gamma}^\top = \mathbf{\Gamma} \mathbf{\Gamma}^+ = (\mathbf{\Gamma}^\top)^+ \mathbf{\Gamma}^\top.$$

Applying the pseudoinverse $(\mathbf{\Gamma}^\top)^+$ to the decision-boundary normal $\mathbf{v}_\epsilon$ gives

$$\tilde{\mathbf{v}}_\epsilon := (\mathbf{\Gamma}^\top)^+ \mathbf{v}_\epsilon.$$

This maps the input-space decision-boundary normal to the visible subspace of the target layer's input. Similarly, we project each signature $\alpha_j \mathbf{w}_j$ onto the same visible subspace:

$$\alpha_j \tilde{\mathbf{w}}_j := P_{\mathbf{\Gamma}} \alpha_j \mathbf{w}_j.$$

The $\mathbf{\Gamma}$ -weighted cosine method uses $z_\epsilon^{(\mathbf{\Gamma})} = \cos(\alpha_t \tilde{\mathbf{w}}_t, \tilde{\mathbf{v}}_\epsilon)$, which removes the singular-value-dependent expansion and contraction induced by $\mathbf{\Gamma}^\top$ in the input space and measures cosine in the visible subspace of the target layer's input.

Model 1 ($\mathbf{\Gamma}$ -weighted cosine) *We assume that a $\mathbf{\Gamma}$ -weighted cosine follows*

$$z_{\text{off}}^{(\mathbf{\Gamma})} \stackrel{\text{approx}}{\sim} \mathcal{N}\left(0, \frac{1}{r_{\mathbf{\Gamma}}}\right), \quad z_{\text{on}}^{(\mathbf{\Gamma})} \stackrel{\text{approx}}{\sim} \mathcal{N}\left(0, \frac{1}{r_{\mathbf{\Gamma}}} + \frac{1}{|\mathcal{J}_{\text{act}}|}\right),$$

where $\mathcal{J}_{\text{act}}$ is the set of active neurons contributing to the off-side normal.

The off-side model is simply derived from an assumption that $\alpha_t \tilde{\mathbf{w}}_t$ and $\tilde{\mathbf{v}}_{\text{off}}$ are independent and isotropic in direction in the $r_{\mathbf{\Gamma}}$ -dimensional visible subspace.

In the on-side model, only the activation state of the target neuron changes. A $\tilde{\mathbf{w}}_t$ component is added to the off-side decision-boundary normal $\tilde{\mathbf{v}}_{\text{off}}$.

$$\tilde{\mathbf{v}}_{\text{on}} = \tilde{\mathbf{v}}_{\text{off}} + \beta_t \tilde{\mathbf{w}}_t = \sum_{j \in \mathcal{J}_{\text{act}}} \beta_j \tilde{\mathbf{w}}_j + \beta_t \tilde{\mathbf{w}}_t.$$

Thus, the on-side cosine is

$$z_{\text{on}}^{(\mathbf{\Gamma})} = \cos(\alpha_t \tilde{\mathbf{w}}_t, \tilde{\mathbf{v}}_{\text{on}}) = \frac{\alpha_t \tilde{\mathbf{w}}_t^\top \tilde{\mathbf{v}}_{\text{off}} + \alpha_t \tilde{\mathbf{w}}_t^\top \beta_t \tilde{\mathbf{w}}_t}{\|\tilde{\mathbf{w}}_t\| \|\tilde{\mathbf{v}}_{\text{on}}\|}.$$

If the target component is sufficiently small compared with the off-side background, we approximate $\|\tilde{\mathbf{v}}_{\text{on}}\| \approx \|\tilde{\mathbf{v}}_{\text{off}}\|$. Then,

$$z_{\text{on}}^{(\mathbf{\Gamma})} \approx z_{\text{off}}^{(\mathbf{\Gamma})} + \frac{\alpha_t \beta_t \|\tilde{\mathbf{w}}_t\|}{\|\tilde{\mathbf{v}}_{\text{off}}\|} = z_{\text{off}}^{(\mathbf{\Gamma})} + \frac{\alpha_t \beta_t \|\tilde{\mathbf{w}}_t\|}{\left\| \sum_{j \in \mathcal{J}_{\text{act}}} \beta_j \tilde{\mathbf{w}}_j \right\|}.$$

In high dimensions with sufficiently many active components, the cross terms between distinct $\tilde{\mathbf{w}}_j$ tend to cancel on average. Therefore, $\left\| \sum_{j \in \mathcal{J}_{\text{act}}} \beta_j \tilde{\mathbf{w}}_j \right\|^2 \approx$

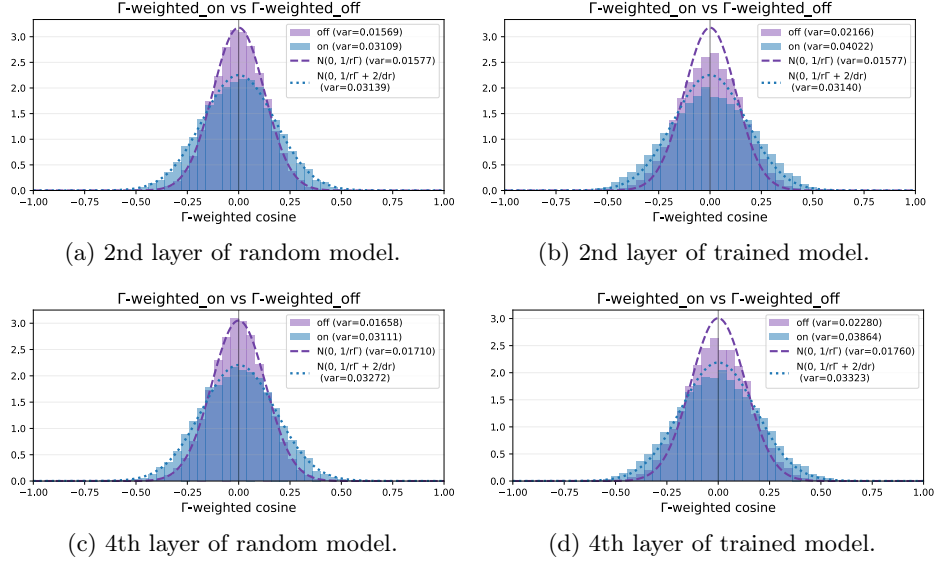


Fig. 7: Γ -weighted cosine distributions for random and MNIST-trained models.

$\sum_{j \in \mathcal{J}_{\text{act}}} \beta_j^2 \|\tilde{\mathbf{w}}_j\|^2$. We further assume that $\tilde{\mathbf{w}}_j$ comes from distributions of comparable scale independently of j . Under this assumption, the variance of $z_{\text{on}}^{(\Gamma)}$ typically becomes

$$\text{Var} \left(z_{\text{on}}^{(\Gamma)} \right) \approx \frac{1}{r_{\Gamma}} + \frac{\beta_t^2}{\sum_{j \in \mathcal{J}_{\text{act}}} \beta_j^2} \approx \frac{1}{r_{\Gamma}} + \frac{1}{|\mathcal{J}_{\text{act}}|},$$

with an assumption that β_j is independently and identically distributed across intersection spaces and j with a mean of zero. As a result, the on-side model is derived by assuming that $z_{\text{on}}^{(\Gamma)}$ follows a normal distribution.

Experiments. Model 1 is derived based on many assumptions. We therefore assess this model empirically, and Fig. 7 summarizes the results. The target model and utilized intersection spaces are exactly the same as those in Fig. 6. Model 1 fits the experimental results well in the initialized model. On the other hand, we still observe a gap in the trained model.

3.4 Signature-Weighted Cosine Method

The weights of a trained model cannot be assumed to be random. Multiple neuron weights may be trained to concentrate in the same direction. If the projected target weight $\tilde{\mathbf{w}}_t$ is relatively aligned with the other $\tilde{\mathbf{w}}_j$, then, because $\tilde{\mathbf{v}}_{\text{off}} = \sum_{j \in \mathcal{J}_{\text{act}}} \beta_j \tilde{\mathbf{w}}_j$, the cosine similarity between $\tilde{\mathbf{v}}_{\text{off}}$ and $\tilde{\mathbf{w}}_t$ has a large variance. As a result, distinguishing $\tilde{\mathbf{v}}_{\text{on}}$ from $\tilde{\mathbf{v}}_{\text{off}}$ becomes difficult.

To remove this anisotropy, we compare cosine similarities in a space whitened using the projected recovered signatures $\alpha_j \tilde{\mathbf{w}}_j$. This whitening suppresses contributions from dense directions and evaluates contributions from rare directions more strongly. Ideally, the competitor space should be formed by recovered signatures of active non-target neurons. However, the attacker cannot determine which neuron is active without knowing each sign. Therefore, the competitor space is formed by the recovered signatures of all non-target neurons instead. We assume that the distribution of directions of all non-target signatures approximately reflects the directional bias produced by real active competitors. We call this method the *signature-weighted cosine method*.

Again, we use $\alpha_j \tilde{\mathbf{w}}_j = P_{\mathbf{r}} \alpha_j \mathbf{w}_j$ and $\tilde{\mathbf{v}}_\epsilon = (\mathbf{\Gamma}^\top)^+ \mathbf{v}_\epsilon$. Let $\mathcal{J}_{\text{comp}} := \{1, \dots, d_\ell\} \setminus \{t\}$ be the index set of neurons in the same layer other than the target neuron. Let

$$\tilde{\mathbf{C}} := \sum_{j \in \mathcal{J}_{\text{comp}}} (\alpha_j \tilde{\mathbf{w}}_j)(\alpha_j \tilde{\mathbf{w}}_j)^\top = \sum_{j \in \mathcal{J}_{\text{comp}}} \tilde{\mathbf{w}}_j \tilde{\mathbf{w}}_j^\top.$$

Note that the correction matrix $\tilde{\mathbf{C}}$ constructed from the recovered signatures is identical to that constructed from the true weights. Then, the signature-weighted cosine is defined by

$$z_\epsilon := \cos_{\tilde{\mathbf{C}}}(\alpha_t \tilde{\mathbf{w}}_t, \tilde{\mathbf{v}}_\epsilon) := \cos(\alpha_t \tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{w}}_t, \tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_\epsilon),$$

where we consider the case $\text{rank}(\tilde{\mathbf{C}}) = r_{\mathbf{r}}$ and interpret $\tilde{\mathbf{C}}^{-1/2}$ as the inverse square root restricted to $\text{Im}(P_{\mathbf{r}})$. Note that when it is rank-deficient, we can use another strategy, a *projection-based method* in Section 3.6.

Model 2 (Signature-weighted cosine) *We assume that a signature-weighted cosine follows*

$$z_{\text{off}} \stackrel{\text{approx}}{\sim} \mathcal{N}\left(0, \frac{1}{r_{\mathbf{r}}}\right), \quad z_{\text{on}} \stackrel{\text{approx}}{\sim} \mathcal{N}\left(0, \frac{1}{r_{\mathbf{r}}} + \frac{\kappa}{|\mathcal{J}_{\text{act}}|}\right).$$

where $\kappa = \frac{|\mathcal{J}_{\text{comp}}|}{|\mathcal{J}_{\text{comp}}| - r_{\mathbf{r}} - 1}$ with $|\mathcal{J}_{\text{comp}}| > r_{\mathbf{r}} + 1$.

Applying $\tilde{\mathbf{C}}^{-1/2}$ is standard whitening that removes this anisotropy. It suppresses directions in which competitors are dense, while strengthening directions in which competitors are sparse. As a result, $\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{off}}$ approaches an approximately isotropic vector in $r_{\mathbf{r}}$ dimensions.

The purpose of the signature-weighted cosine method is to remove the direction-dependent bias of $\tilde{\mathbf{v}}_{\text{off}}$. However, it also relatively emphasizes the target component added on the on side. This is because $\tilde{\mathbf{C}}$ is constructed only from competitors and does not include $\tilde{\mathbf{w}}_t$.

In a local region where only the target ReLU switches,

$$\tilde{\mathbf{v}}_{\text{on}} = \tilde{\mathbf{v}}_{\text{off}} + \beta_t \tilde{\mathbf{w}}_t, \quad \tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{on}} = \tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{off}} + \beta_t \tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{w}}_t.$$

Thus,

$$z_{\text{on}} = \frac{(\alpha_t \tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{w}}_t)^\top (\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{off}}) + \alpha_t \beta_t \|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{w}}_t\|^2}{\|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{w}}_t\| \|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{on}}\|}.$$

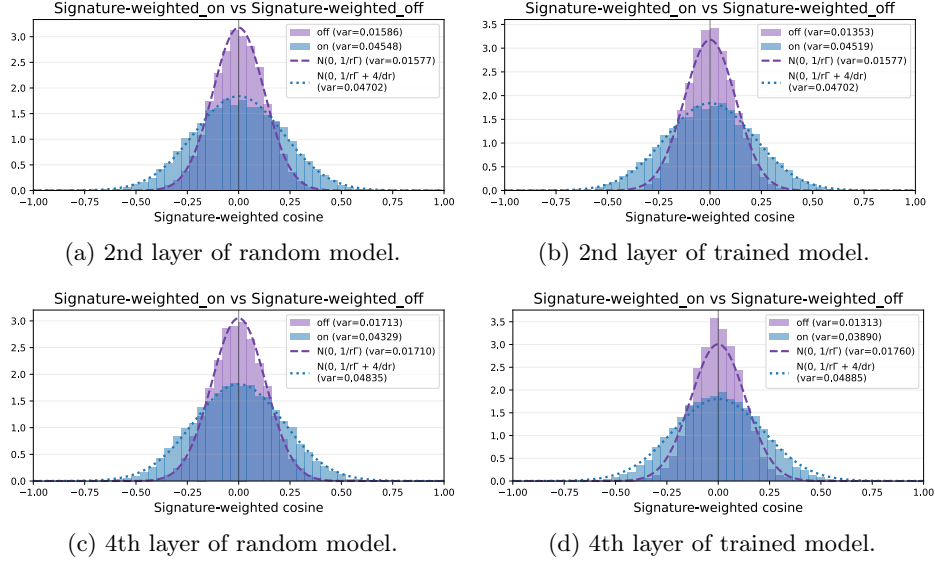


Fig. 8: Signature-weighted cosine distributions for random and MNIST-trained models.

Assuming that the target component is sufficiently small relative to the off-side background so that $\|\tilde{\mathbf{C}}^{-1/2}\tilde{\mathbf{v}}_{\text{on}}\| \approx \|\tilde{\mathbf{C}}^{-1/2}\tilde{\mathbf{v}}_{\text{off}}\|$, we obtain

$$z_{\text{on}} \approx z_{\text{off}} + \frac{\alpha_t \beta_t \|\tilde{\mathbf{C}}^{-1/2}\tilde{\mathbf{w}}_t\|}{\|\tilde{\mathbf{C}}^{-1/2}\tilde{\mathbf{v}}_{\text{off}}\|}.$$

This additional term is amplified from $\alpha_t \beta_t \|\tilde{\mathbf{w}}_t\|/\|\tilde{\mathbf{v}}_{\text{off}}\|$ in the $\mathbf{\Gamma}$ -weighted cosine. As derived in Section D, the typical squared magnitude is estimated as

$$\left(\frac{\alpha_t \beta_t \|\tilde{\mathbf{C}}^{-1/2}\tilde{\mathbf{w}}_t\|}{\|\tilde{\mathbf{C}}^{-1/2}\tilde{\mathbf{v}}_{\text{off}}\|} \right)^2 \approx \frac{|\mathcal{J}_{\text{comp}}|}{|\mathcal{J}_{\text{comp}}| - r_{\mathbf{\Gamma}} - 1} \frac{\beta_t^2}{\sum_{j \in \mathcal{J}_{\text{act}}} \beta_j^2} \approx \frac{|\mathcal{J}_{\text{comp}}|}{|\mathcal{J}_{\text{comp}}| - r_{\mathbf{\Gamma}} - 1} \frac{1}{|\mathcal{J}_{\text{act}}|}.$$

Experiments. Fig. 8 summarizes the results of the signature-weighted cosine method. Again, the target model and utilized intersection spaces are exactly the same as those in Figs. 6 and 7. Using the signature-weighted cosine method produces results closer to the expected model than the $\mathbf{\Gamma}$ -weighted cosine method. Besides, we can observe that the cosine distributions for the on side yield larger variance.

3.5 Algorithm and Estimating the Number of Intersection Spaces

Algorithm 1 shows the pseudocode for sign recovery. Each intersection space yields two values, z_i and z'_i . The value aggregated for sign classification is $D_i :=$

Algorithm 1 SIGNATUREWEIGHTEDSIGNRECOVERY

Require: N intersection spaces; target $\alpha_t \mathbf{w}_t$; competitors $\{\alpha_j \mathbf{w}_j\}_{j \neq t}$; recovered preceding-layer weights and biases

Ensure: the sign information for the target neuron

- 1: $\mathcal{D} \leftarrow \emptyset$
 - 2: **for all** given intersection space $(\mathbf{s}, \mathbf{v}, \mathbf{v}')$ **do**
 - 3: Obtain adjacent boundary points $\mathbf{x}$ and $\mathbf{x}'$.
 - 4: $\mathbf{\Gamma} \leftarrow \mathbf{\Gamma}_s^{(\ell-1)}$.
 - 5: **if** $\alpha_t \mathbf{w}_t^\top \mathbf{\Gamma}(\mathbf{x} - \mathbf{s}) < 0$ **then**
 - 6: $(\mathbf{v}, \mathbf{v}') \leftarrow (\mathbf{v}', \mathbf{v})$.
 - 7: $(\tilde{\mathbf{v}}, \tilde{\mathbf{v}}') \leftarrow ((\mathbf{\Gamma}^\top)^+ \mathbf{v}, (\mathbf{\Gamma}^\top)^+ \mathbf{v}')$.
 - 8: $\alpha_j \tilde{\mathbf{w}}_j \leftarrow \Pr(\alpha_j \mathbf{w}_j)$ for $j = 1, \dots, d_\ell$.
 - 9: $\tilde{\mathbf{C}} \leftarrow \sum_{j \in \mathcal{J}_{\text{comp}}} (\alpha_j \tilde{\mathbf{w}}_j)(\alpha_j \tilde{\mathbf{w}}_j)^\top$.
 - 10: $(z, z') \leftarrow (\cos_{\tilde{\mathbf{C}}}(\alpha_t \tilde{\mathbf{w}}_t, \tilde{\mathbf{v}}), \cos_{\tilde{\mathbf{C}}}(\alpha_t \tilde{\mathbf{w}}_t, \tilde{\mathbf{v}}'))$.
 - 11: Append $z^2 - z'^2$ to $\mathcal{D}$.
 - 12: $T \leftarrow \frac{\sqrt{N} \overline{\mathcal{D}}}{s_{\mathcal{D}}}$, where $\overline{\mathcal{D}}$ and $s_{\mathcal{D}}$ are the sample mean and standard deviation of $\mathcal{D}$.
 - 13: **if** $T < 0$ **then return** $\alpha_t = -1$.
 - 14: **return** $\alpha_t = 1$.
-

$z_i^2 - z_i'^2$. This score directly measures the difference in their variances when their means can be approximated as zero. To estimate the required number of intersection spaces, we regard both sides as zero-mean normal distributions. Under an approximation that ignores covariance,

$$\mathbb{E}[D_i] = \text{Var}(z) - \text{Var}(z'), \quad \text{Var}(D_i) \simeq 2 \text{Var}(z)^2 + 2 \text{Var}(z')^2.$$

Therefore, the expected T value for N samples is

$$T \simeq \sqrt{N} \frac{\text{Var}(z) - \text{Var}(z')}{\sqrt{2 \text{Var}(z)^2 + 2 \text{Var}(z')^2}}.$$

Thus, when $T > 0$, $\mathbf{v}$ is the on-side. When $T < 0$, $\mathbf{v}'$ is the on side.

For the signature-weighted cosine method, the additional on-side term is about $\frac{|\mathcal{J}_{\text{comp}}|}{|\mathcal{J}_{\text{comp}}| - r_{\mathbf{\Gamma}} - 1} \frac{1}{|\mathcal{J}_{\text{act}}|} \approx 4/d_\ell$ when $|\mathcal{J}_{\text{comp}}| = d_\ell - 1$ and $|\mathcal{J}_{\text{act}}| \approx r_{\mathbf{\Gamma}} \approx d_\ell/2$.

$$\text{Var}(z_{\text{off}}) \simeq \frac{1}{r_{\mathbf{\Gamma}}} \simeq \frac{2}{d_\ell}, \quad \text{Var}(z_{\text{on}}) \simeq \frac{2}{d_\ell} + \frac{4}{d_\ell} = \frac{6}{d_\ell}.$$

Then, $\mathbb{E}[D_i] \simeq 4/d_\ell$ and $\text{Var}(D_i) \simeq 80/d_\ell^2$, and $N \gtrsim 45$ yields $|T| \geq 3$ under the independent normal approximation assumption.

3.6 Correcting Unreliable Signs

The signature-weighted cosine method is based on empirical statistical bias. Since the underlying assumption does not always hold in trained models, we need to verify the recovered signs and correct the overall result if necessary. For this purpose, we introduce two methods.

Active-Weighted Cosine Method. As mentioned in Section 3.4, ideally, the competitor space should be formed by active weights $\mathcal{J}_{\text{act}}$. Actually, the signature-weighted cosine method also whitens the weights of neurons that are actually inactive and do not contribute to $\tilde{\mathbf{v}}_{\text{off}}$. Therefore, it is reasonable to fix highly reliable neuron signs, e.g., $|t| \geq 3$ in the signature-weighted cosine method, and remove these weights from the competitor space if they are inactive in each intersection space. Let $\mathcal{J}_{\text{cand}} := \mathcal{J}_{\text{comp}} \setminus \mathcal{J}_{\text{inactive}}$, where $\mathcal{J}_{\text{inactive}}$ contains indices whose neurons are identified as inactive according to highly reliable signs. Then,

$$\tilde{\mathbf{C}}_{\text{cand}} := \sum_{j \in \mathcal{J}_{\text{cand}}} (\alpha_j \tilde{\mathbf{w}}_j)(\alpha_j \tilde{\mathbf{w}}_j)^\top = \sum_{j \in \mathcal{J}_{\text{cand}}} \tilde{\mathbf{w}}_j \tilde{\mathbf{w}}_j^\top.$$

When $\tilde{\mathbf{C}}_{\text{cand}}$ is invertible on $\text{Im}(\mathbf{P}_{\mathbf{r}})$, we use $\cos_{\tilde{\mathbf{C}}_{\text{cand}}}(\alpha_t \tilde{\mathbf{w}}_t, \tilde{\mathbf{v}}_\epsilon)$. To distinguish between the terms, we call this method the *active-weighted cosine method*. The rank-deficient case is handled by the projection-based method described below.

Projection-Based Method. After recovering many signs, we may obtain an intersection space satisfying $\text{rank}(\tilde{\mathbf{C}}_{\text{cand}}) < r_{\mathbf{r}}$. Then, we can use a powerful analytic method.

Proposition 1. *Suppose that $\mathcal{J}_{\text{act}} \subseteq \mathcal{J}_{\text{cand}}$ and $\text{rank}(\tilde{\mathbf{C}}_{\text{cand}}) < r_{\mathbf{r}}$. Then, the off-side normal $\tilde{\mathbf{v}}_{\text{off}}$ lies in the competitor span $\text{Im}(\tilde{\mathbf{C}}_{\text{cand}})$. Unless $\tilde{\mathbf{w}}_t$ happens to lie entirely in the competitor span, projection onto the competitor span leaves the off-side normal unchanged and changes the on-side normal. Thus, the sign can be determined directly from a single intersection space usually.*

Proposition 1 implies that the sign is easily recovered once $\text{rank}(\tilde{\mathbf{C}}_{\text{cand}}) < r_{\mathbf{r}}$ holds. Let's consider the model using $784 - 128 - \dots$ and try to recover the signs in the 1st layer. Then, $\text{rank}(\tilde{\mathbf{C}}_{\text{cand}}) \leq 127$ and $r_{\mathbf{r}} = 784$, and this condition always holds. Thus, we can recover all the signs in the first layer.

For subsequent layers, this condition does not hold initially because we cannot identify which neurons are active. However, after recovering and fixing the signs of almost all neurons, some intersection spaces could satisfy this condition. Once we find it, we can determine all signs. Specifically, all signs identified as inactive neurons are correct because $\tilde{\mathbf{v}}_{\text{off}}$ lies in the competitor span. For neurons identified as active, we remove such a neuron from the competitor span one by one and verify whether $\tilde{\mathbf{v}}_{\text{off}}$ still lies in the competitor span. In this way, we can determine the signs of all neurons.

In a noise-free environment, this projection-based method can reliably recover each sign, except in negligible cases such as when β_j is exactly 0 or when a weight can be fully described by other weights. However, in practice, end-to-end extraction always introduces non-negligible computational noise, which means this method is not fully deterministic. This is one of the reasons that the actual end-to-end extraction is significantly harder than sign recovery alone. For details, please refer to Section 4.

Table 2: Sign-recovery results for an MNIST-trained 784-128⁽⁵⁾-10 model from 50 intersection spaces per evaluated neuron. Numbers labeled by HC denote high-confidence results, where $|t| \geq 3$ for the signature-weighted cosine method and $CL \geq 99.7\%$ for the boundary-walking method. Here, HC under Wrong denotes a high-confidence error, namely, a case in which the incorrect sign direction is selected with high confidence.

Layer	Evaluated	Signature-weighted cosine		Boundary-walking	
		Correct (HC)	Wrong (HC)	Correct (HC)	Wrong (HC)
2	128	127 (119)	1 (0)	105 (0)	23 (0)
3	128	128 (110)	0 (0)	85 (0)	43 (0)
4	128	128 (120)	0 (0)	80 (0)	48 (0)
5	126	123 (107)	3 (0)	67 (0)	59 (0)

3.7 Comparison with the Existing Method

We compare the accuracy of sign recovery by the signature-weighted cosine method and the existing boundary-walking method using the same intersection spaces. The experiment is conducted in a setting where intersection spaces, target-layer signatures, past-layer weights, and past-layer biases are given without noise. Under such a setting, our method does not require additional dedicated queries. In contrast, the boundary-walk method requires dedicated queries, and we evaluated it using a white-box implementation, as in prior work. The target model is an MNIST-trained 784-128⁽⁵⁾-10 model. We used 50 intersection spaces for each evaluated neuron, excluding persistent and dead neurons.

Table 2 summarizes the sign-recovery result for each layer of this target model. The boundary-walking method did not yield any high-confidence results, indicating that 50 intersection spaces are insufficient. This is not surprising, as prior work has already reported that about 10^2 - 10^3 intersection spaces are required [2]. In contrast, the signature-weighted cosine method recovers the signs of almost all neurons with a high confidence level.

Although the signature-weighted cosine method performs much better than the existing method, opposite signs are wrongly recovered in a few neurons, e.g., one and three neurons in the 2nd and 5th layers. As discussed in Section 3.6, the projection-based method can identify or correct additional signs when the required span conditions hold. In fact, we could verify and correct all signs of this model with the projection-based method. If the projection-based method fails to verify, we can proceed to the active-weighted cosine method, in which inactive neurons whose signs are recovered with high reliability are excluded from the competitor.

4 End-to-End Extraction

This section addresses the challenges of end-to-end hard-label model extraction and presents methods for overcoming them. Recovering parameters in deeper

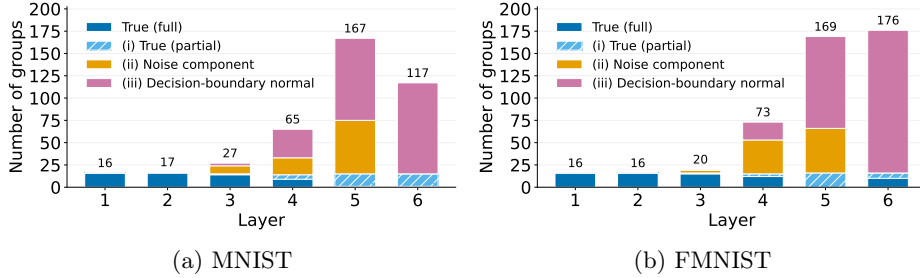


Fig. 9: Groups passing the consistency check by layer, using 20,000 intersection spaces per trained model with six hidden layers of width 16.

layers introduces difficulties that do not arise in shallow layers: numerical errors in previously recovered weights accumulate, and activation patterns in subsequent layers exhibit limited variation. These issues also affect the sign-recovery algorithms presented in Section 3, which must tolerate such errors to support end-to-end extraction. We first describe the challenges in signature recovery and sign recovery and present our solutions. We then evaluate the resulting end-to-end attack on models with four or six hidden layers of width 16.

We do not evaluate recovery of persistent neurons, which remain active for every input, because none were identified in our experiments. Existing cross-layer extraction techniques [8] could accommodate such neurons, but we do not evaluate this extension.

4.1 Improving Signature Recovery

We address three challenges in signature recovery for end-to-end extraction. (i) Some weights are difficult to recover because the corresponding neurons in the preceding layer are rarely or never active on the target neuron’s activation boundary; these are called *query-intensive weights* and *unreachable weights*. (ii) intersection spaces associated with neurons in subsequent layers can pass the consistency check for the target layer, producing *noise components*. (iii) intersection spaces with shared activation patterns in subsequent layers can cause a decision-boundary normal to be recovered as a signature. Issues (i) and (ii) also arise in soft-label extraction [10], whereas (iii) is specific to hard labels.

Fig. 9 shows the breakdown of groups passing the consistency check among 20,000 intersection spaces collected from each of two models with six hidden layers of width 16, trained on MNIST and FMNIST, respectively. True (full) and True (partial) denote groups corresponding to genuine signatures for which all coordinates or only a subset of coordinates can be recovered, respectively. True (partial) is associated with issue (i), while Noise component and Decision-boundary normal correspond to issues (ii) and (iii), respectively. We identified True (full) and True (partial) groups through white-box analysis using the ground-truth weights of each layer. Their combined count can fall below the

layer width of 16 because 20,000 intersection spaces may be insufficient or because some neurons are dead. The end-to-end experiments in Section 4.3 use 400,000 intersection spaces for each model with six hidden layers. As the figure illustrates, filtering out groups arising from issues (ii) and (iii) should essentially leave only those corresponding to genuine signatures of the target layer. The missing coordinates of True (partial) signatures must then be recovered. We describe each issue and our solution under the corresponding heading (i)–(iii) below. When discussing signature recovery, we use $\mathbf{w}$ for a recovered signature candidate whose sign has yet to be determined, omitting the explicit factor α .

(i) Query-Intensive and Unreachable Weights. As observed in [10], dependencies between the activation patterns of neurons in adjacent layers can make certain weights difficult to recover. The same issue arises in the hard-label setting. Consider recovering the signature of the k th neuron in layer ℓ , whose weight vector is $\mathbf{w}_k^{(\ell)}$. If the j th neuron in layer $\ell - 1$ is usually or always inactive on the target neuron’s activation boundary, the j th input coordinate to layer ℓ is correspondingly usually or always zero at the target neuron’s intersection points. Points at which this coordinate is zero do not provide information about the missing signature coordinate $w_{k,j}^{(\ell)}$.

In [10], a weight is called *query-intensive* if the preceding neuron is active with very low probability on the target neuron’s activation boundary, requiring a large number of queries for recovery. An effective weight is called *unreachable* if the preceding neuron is never active on the target activation boundary, yet the two neurons can be active together away from that boundary. A weight is non-effective if the two neurons cannot be active together. Effective query-intensive and unreachable weights cannot simply be ignored: the preceding neuron need not be dead and may be active when the target neuron is also active. Indeed, the experiments in [10] show that unreachable weights reduce layer coverage, particularly in deeper layers.

We use two approaches to address this issue. The first adaptively queries the model to search for intersection points that reveal the missing coordinate. It applies when such points exist but are difficult to find, as in the case of query-intensive weights. The second uses cross-layer extraction [8] to recover the missing coordinate from activation boundaries in the next layer. This enables recovery even without a suitable intersection point for the target neuron.

Searching for informative intersection points. Starting from an existing intersection point of the target neuron, we move along the intersection of its activation boundary and the decision boundary toward a region where the preceding neuron associated with the missing coordinate becomes active. We then perturb the input into that region and search nearby for a new intersection point that reveals the missing coordinate. The detailed search procedure, including direction updates and restarts, is given in Section G.1.

Cross-layer extraction. The first approach cannot recover unreachable weights, so we instead use cross-layer extraction [8]. This technique was originally pro-

posed to recover the weights of *persistent neurons*, which remain active for every input. For such neurons, ReLU acts as the identity, allowing their weights to be combined with those of the next layer. Their combined contributions can then be estimated using activation boundaries in that layer.

We apply this principle to unreachable weights; see Section F.

(ii) Spurious Recovery of Subsequent-Layer Signatures. In soft-label extraction, incorrectly grouping critical points associated with subsequent-layer neurons produces *noise components* [10]. The same issue arises in consistency checks on intersection spaces in the hard-label setting.

Suppose that the attacker is performing consistency checks for layer ℓ . Let $\mathbf{s}_1$ and $\mathbf{s}_2$ be two intersection points associated with the k th neuron in layer $\ell + 1$, whose weight vector and bias are $\mathbf{w}_k^{(\ell+1)}$ and $b_k^{(\ell+1)}$. Suppose that layer ℓ has the same activation pattern at both points, represented by the diagonal matrix $\mathbf{D}^{(\ell)}$. Let $\mathbf{h}_i^{(\ell-1)} = \mathbf{h}^{(\ell-1)}(\mathbf{s}_i)$ be the input to layer ℓ at $\mathbf{s}_i$, for $i \in \{1, 2\}$. Since both points lie on the same next-layer activation boundary,

$$\begin{cases} \mathbf{w}_k^{(\ell+1)\top} \mathbf{D}^{(\ell)} \mathbf{W}^{(\ell)} \mathbf{h}_1^{(\ell-1)} + \mathbf{w}_k^{(\ell+1)\top} \mathbf{D}^{(\ell)} \mathbf{b}^{(\ell)} + b_k^{(\ell+1)} = 0, \\ \mathbf{w}_k^{(\ell+1)\top} \mathbf{D}^{(\ell)} \mathbf{W}^{(\ell)} \mathbf{h}_2^{(\ell-1)} + \mathbf{w}_k^{(\ell+1)\top} \mathbf{D}^{(\ell)} \mathbf{b}^{(\ell)} + b_k^{(\ell+1)} = 0. \end{cases}$$

Define $\tilde{\mathbf{w}}_k^{(\ell+1)} = \mathbf{W}^{(\ell)\top} \mathbf{D}^{(\ell)\top} \mathbf{w}_k^{(\ell+1)}$, $\tilde{b}_k^{(\ell+1)} = \mathbf{w}_k^{(\ell+1)\top} \mathbf{D}^{(\ell)} \mathbf{b}^{(\ell)} + b_k^{(\ell+1)}$. Then both points satisfy $\tilde{\mathbf{w}}_k^{(\ell+1)\top} \mathbf{h}_i^{(\ell-1)} + \tilde{b}_k^{(\ell+1)} = 0$, $i \in \{1, 2\}$. This constraint holds not only at the representative points but throughout each corresponding intersection space after applying the local affine map to the input space of layer ℓ . When $\tilde{\mathbf{w}}_k^{(\ell+1)}$ is nonzero, the mapped spaces lie in a common hyperplane and can pass the consistency check for layer ℓ .

The same issue can arise even when the j th neuron in layer ℓ changes its activation state across the collected intersection points. Suppose that each point corresponds to the k th neuron in layer $\ell + 1$, while the state of the j th neuron in layer ℓ is fixed in a neighborhood of each point. The class pair defining the decision boundary and the activation patterns of the other neurons from layer ℓ onward, excluding the boundary neuron k in layer $\ell + 1$, are shared, whereas activation patterns in preceding layers may vary freely. The corresponding intersection spaces can then pass the consistency check even when the j th neuron in layer ℓ has different activation states. Its output appears linearly in both the next-layer neuron's activation-boundary equation and the decision-boundary equation, so its contribution can be eliminated by taking a linear combination of the two equations. Here, the intersection spaces for both activation states, mapped to the input space of layer ℓ , lie in a common hyperplane. Section E derives this shared constraint, including the decision-boundary coefficients obtained from the network formed by subsequent layers. We also call this a *noise component*, since it results from accepting subsequent-layer intersection spaces as those of a target-layer neuron.

Solution. To filter out noise components in both cases, we check whether each recovered candidate is an actual target-layer activation boundary. In the soft-label setting, noise components can be rejected by checking whether the activation boundary bends [10]. This test cannot be applied directly in the hard-label setting, where only class labels are observable. Instead, we search in other ReLU cells for intersections between the candidate activation boundary and the decision boundary, and check whether the latter bends at these intersections. A noise component may define a hyperplane containing the mapped intersection spaces without corresponding to an actual activation boundary of the target layer. Such a candidate need not induce a bend in the decision boundary at a new intersection, whereas a genuine activation boundary is expected to do so. We use this distinction to validate candidate signatures empirically.

We start from collected intersection points that were not used to recover the candidate and move along an adjacent decision-boundary toward the candidate activation boundary. We prioritize starting points with shorter predicted travel distances to reduce the likelihood of crossing other ReLU cells. Section G.2 gives the detailed procedure and the displacement calculation.

(iii) Spurious Recovery of Decision-Boundary Normals. Limited variation in activation patterns in deeper layers underlies problems such as the noise components reported in [10]. Even random inputs may produce only a limited set of activation patterns in subsequent layers. Viewed from the input space of an intermediate layer, fewer ReLU cells are encountered, and activation boundaries subdivide the decision boundary less frequently. As a result, multiple intersection spaces, mapped to that input space, are more likely to lie in a shared decision-boundary hyperplane. They can therefore pass the consistency check, yielding the hyperplane normal and offset as a spurious signature and bias.

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two intersection spaces whose images under the respective local affine maps to the input space of layer ℓ lie in a shared decision-boundary hyperplane. Let $\mathbf{s}_1 \in \mathcal{S}_1$ and $\mathbf{s}_2 \in \mathcal{S}_2$ be their representative intersection points. Let $\mathbf{v}^{(\ell)}$ and $b^{(\ell)}$ be the normal and offset of this hyperplane, respectively. Then $\mathbf{v}^{(\ell)\top} \mathbf{h}^{(\ell-1)}(\mathbf{s}_i) + b^{(\ell)} = 0$, $i \in \{1, 2\}$. This shared hyperplane can cause $\mathbf{v}^{(\ell)}$ to be incorrectly recovered as a signature.

These points may nevertheless appear adjacent to different hyperplanes in the model’s input space. Let $\mathbf{\Gamma}_1^{(\ell-1)}$ and $\mathbf{\Gamma}_2^{(\ell-1)}$ be the forward local linear matrices through layer $\ell - 1$ at $\mathbf{s}_1$ and $\mathbf{s}_2$, respectively. The corresponding input-space normals are $\mathbf{v}_1^{(1)} = \mathbf{\Gamma}_1^{(\ell-1)\top} \mathbf{v}^{(\ell)}$, $\mathbf{v}_2^{(1)} = \mathbf{\Gamma}_2^{(\ell-1)\top} \mathbf{v}^{(\ell)}$. Activation patterns in preceding layers typically differ between the two points, yielding different local linear maps and potentially different input-space normals. Thus, the shared decision boundary may not be apparent in the model’s input space.

To detect such spurious signatures, we pull the recovered signature back to the input space at the representative point of each intersection space in a consistent group and compare it with the two adjacent decision-boundary normals. Let $\mathbf{s}_1, \dots, \mathbf{s}_m$ be the intersection points of the spaces in the group, let $(\mathbf{v}_i, \mathbf{v}'_i)$ be the pair of normals associated with $\mathbf{s}_i$, and let $\mathbf{w}$ be the recovered signature. Let $\mathbf{\Gamma}_i$

be the forward local linear matrix to the target layer’s input at $\mathbf{s}_i$. If $\mathbf{w}$ is actually a shared decision-boundary normal in that space, its pullback should be parallel to one of the observed normals at every point: $\mathbf{v}_i \parallel \mathbf{\Gamma}_i^\top \mathbf{w}$ or $\mathbf{v}'_i \parallel \mathbf{\Gamma}_i^\top \mathbf{w}$, $i \in [m]$.

For numerical robustness, we measure the fraction of aligned points in each group. For each point, we compute the maximum absolute cosine similarity $c_i = \max(|\cos(\mathbf{\Gamma}_i^\top \mathbf{w}, \mathbf{v}_i)|, |\cos(\mathbf{\Gamma}_i^\top \mathbf{w}, \mathbf{v}'_i)|)$. We reject a group as a suspected decision-boundary component if $\frac{\#\{i \in [m] | c_i < \rho\}}{m} < \tau$. We use $\rho = 0.97$ and $\tau = 0.01$ throughout, as these gave the best experimental results.

4.2 Stabilizing Sign Recovery

In Section 3, we demonstrated that most signs can be recovered from a small number of intersection spaces when accurate signatures and preceding-layer parameters are available. In end-to-end extraction, numerical errors and incomplete recovery make reliable sign recovery more challenging. We therefore use a two-stage procedure that empirically improves robustness: first, we apply the methods of Section 3 to assign signs in order of confidence, incorporating each assignment into the remaining estimates; then, we check the assignments against the collected decision-boundary normals and correct them when necessary.

Step 1: Provisional Sign Assignment. For each neuron with an unassigned sign, we apply a projection- or cosine-based method from Section 3 to its intersection spaces. Following Section 3, let $\mathcal{J}_{\text{comp}}$ be the index set of recovered non-target neurons and $\mathcal{J}_{\text{inactive}}$ the subset classified as inactive at the current point under the assigned signs. We form $\mathcal{J}_{\text{cand}} = \mathcal{J}_{\text{comp}} \setminus \mathcal{J}_{\text{inactive}}$ and compare the dimension of its signature span, pulled back to the input space, with the rank of the local linear map. If the former is smaller than the latter and exactly one normal lies in the competitor span, the projection-based method estimates that normal as the off-side normal. This can hold without known signs, often in the first layer and sometimes in the second.

When the projection-based method cannot determine any additional signs, we turn to cosine-based estimation. If no signs are known, $\mathcal{J}_{\text{inactive}} = \emptyset$, so we use the signature-weighted cosine method with $\mathcal{J}_{\text{cand}} = \mathcal{J}_{\text{comp}}$. Once some signs are known, we use the active-weighted cosine method with the updated $\mathcal{J}_{\text{cand}}$. We evaluate signs using the statistic T and the sign decision rule in Algorithm 1. We process the intersection spaces for each neuron sequentially and fix a neuron’s sign as soon as it satisfies $|T| \geq 3$. Each time a single sign is fixed, we update $\mathcal{J}_{\text{inactive}}$ and $\mathcal{J}_{\text{cand}}$ at each point and immediately return to the projection-based method for the remaining neurons. When that method can no longer determine additional signs, we resume cosine-based estimation with the updated competitors. If no remaining neuron reaches the threshold using the available intersection spaces, we provisionally assign the remaining signs according to their respective T statistics. Thus, we prioritize reliable assignments, then check the complete assignment in Step 2.

Table 3: Neuron recovery, queries, and running times for MNIST- and FMNIST-trained models with four or six hidden layers of width 16 (Int.: intersection).

Model	Metric	Int. space collection	L1	L2	L3	L4	L5	L6	L7	Entire model
MNIST 784-16 ⁽⁴⁾ -10	Active neurons	–	16	16	16	16	10	–	–	74
	Recovered neurons	–	16	16	16	16	10	–	–	74
	Signature time	–	2.95s	4.20s	9.02s	5.15s	3.20s	–	–	25s
	Sign time	–	0.23s	44s	29s	31s	0.23ms	–	–	1m44s
	Total queries	$2^{32.31}$	$2^{17.10}$	$2^{17.10}$	$2^{18.42}$	$2^{17.69}$	0	–	–	$2^{32.31}$
MNIST 784-16 ⁽⁶⁾ -10	Active neurons	–	16	16	15	15	15	16	10	103
	Recovered neurons	–	16	16	15	15	15	16	10	103
	Signature time	–	4.72s	17s	17m14s	41m05s	4m09s	1m42s	1m05s	1h05m36s
	Sign time	–	0.87s	4.89s	6m45s	7m06s	6m06s	6m32s	0.15ms	26m34s
	Total queries	$2^{36.62}$	$2^{17.10}$	$2^{17.10}$	$2^{20.44}$	$2^{21.90}$	$2^{21.75}$	$2^{21.82}$	0	$2^{36.62}$
FMNIST 784-16 ⁽⁴⁾ -10	Active neurons	–	16	16	16	16	10	–	–	74
	Recovered neurons	–	16	16	16	16	10	–	–	74
	Signature time	–	2.99s	5.39s	59s	1m21s	3.23s	–	–	2m32s
	Sign time	–	0.24s	35s	39s	33s	0.14ms	–	–	1m48s
	Total queries	$2^{32.30}$	$2^{17.10}$	$2^{17.11}$	$2^{21.71}$	$2^{22.24}$	0	–	–	$2^{32.30}$
FMNIST 784-16 ⁽⁶⁾ -10	Active neurons	–	16	16	16	15	16	16	10	105
	Recovered neurons	–	16	16	16	15	16	16	10	105
	Signature time	–	4.79s	16s	21s	39m25s	1m07s	4m45s	1m05s	47m04s
	Sign time	–	0.90s	4.45s	10m05s	6m14s	6m34s	7m17s	0.15ms	30m16s
	Total queries	$2^{36.62}$	$2^{17.10}$	$2^{17.09}$	$2^{17.47}$	$2^{21.80}$	$2^{21.26}$	$2^{23.67}$	0	$2^{36.62}$

Step 2: Sign Correction Based on Normal-Span Violations. The signs assigned in Step 1 may be incorrect because of numerical errors and incomplete recovery. We check them against the collected decision-boundary normals: with accurate recovery and correct signs, each normal lies in the span of the signatures classified as active on its corresponding side, after they are pulled back to the input space. We count violations of this condition and iteratively flip the sign that most reduces the count, stopping when no single sign flip improves it. Numerical errors and unrecovered neurons may leave residual violations even with correct signs. Section H gives the definitions and detailed correction procedure.

4.3 Experiments

We evaluate end-to-end hard-label extraction on MNIST- and Fashion-MNIST (FMNIST)-trained models, reporting recovered neuron counts, query counts, running times, and output agreement with the oracle. The target models are fully connected ReLU networks with four or six hidden layers of 16 neurons each, 784 inputs, and 10 outputs. We initialized the weights using Kaiming normal initialization and trained the models for 30 epochs using the Adam optimizer with a learning rate of 10^{-3} . We used cross-entropy loss for 10-class classification.

The attack collects intersection spaces before recovering layers from shallow to deep. We collected 20,000 intersection spaces for each model with four hidden layers and 400,000 for each model with six hidden layers. We report the collection query cost separately. For sign recovery, we modify Algorithm 1 to

use $D = |z| - |z'|$ and divide each competitor $\tilde{\mathbf{w}}_j$ by its input-space pullback norm $\|\mathbf{\Gamma}^\top(\alpha_j \mathbf{w}_j)\|_2$ before forming the correction matrix, omitting zero pullbacks. These changes slightly improved performance.

Table 3 summarizes the results. No persistent neurons were identified in these experiments, so we did not perform their recovery. However, we observed dead neurons, which were always inactive, and almost-dead neurons, whose activation probability was at most 1% under standard Gaussian inputs. The Active neurons rows exclude these neurons; for the output layer, they count all 10 output units, to which ReLU is not applied. Recovered neurons gives the number of neurons successfully recovered. Signature time and Sign time report the running times for signature and sign recovery, respectively. In the Total queries row, Int. space collection reports the cost of collecting intersection spaces, L1–L7 report the additional queries for each layer, and Entire model includes both costs. Sign recovery reuses the collected intersection spaces and requires zero additional queries, so all layer-recovery queries are for signature recovery.

For all four models in the table, we recovered every neuron except dead and almost-dead neurons. The maximum L_2 distance between an estimated hidden-layer signature and the corresponding true weight vector, after matching neurons, resolving the sign, and normalizing each vector, was 9.34×10^{-6} . Initial intersection space collection dominated the query cost; signature recovery required relatively few additional queries.

We measured output agreement as the fraction of 100,000 standard Gaussian inputs drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I}_{784})$ for which the extracted model and oracle predicted the same class. The agreement rates were 98.477% and 99.948% for the MNIST models with four and six hidden layers, respectively, and 100.000% and 99.744% for the corresponding FMNIST models. Numerical errors and unrecovered almost-dead neurons are possible sources of the remaining disagreements.

5 Conclusion

We analyzed obstacles to cryptanalytic hard-label model extraction, focusing on sign recovery and also addressing signature recovery, and proposed methods to overcome them. For sign recovery, we highlighted the implementation challenges and large number of intersection spaces required by the boundary-walking method, and proposed cosine and projection-based methods that use simpler computations and fewer intersection spaces. For signature recovery, we addressed spurious groups accepted by consistency checks and unreachable weights in the hard-label setting. Combining these methods, we demonstrated end-to-end extraction of trained ReLU models with four and six hidden layers of width 16, recovering all neurons except dead and almost-dead neurons and achieving over 98% label agreement on standard Gaussian inputs.

Although our methods limit the additional queries needed for signature and sign recovery, collecting the required intersection spaces still incurs a substantial query cost. Making this collection process more efficient remains an important direction for future work.

Acknowledgments. We acknowledge the use of OpenAI Codex (GPT-5.6) to assist with code implementation, debugging, and code review. All research ideas, experimental design, and analyses were developed and verified by the authors.

References

1. 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. *IACR Cryptol. ePrint Arch.* **2024**, 1580 (2024), <https://eprint.iacr.org/2024/1580>
2. 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: Fehr, S., Fouque, P. (eds.) *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Madrid, Spain, May 4-8, 2025, Proceedings, Part I. LNCS, vol. 15601, pp. 364–396. Springer (2025). https://doi.org/10.1007/978-3-031-91107-1_13
3. Carlini, N., Jagielski, M., Mironov, I.: Cryptanalytic extraction of neural network models. In: Micciancio, D., Ristenpart, T. (eds.) *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference*, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. LNCS, vol. 12172, pp. 189–218. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_7
4. Chen, Y., Dong, X., Guo, J., Shen, Y., Wang, A., Wang, X.: Hard-label cryptanalytic extraction of neural network models. In: Chung, K., Sasaki, Y. (eds.) *Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security*, Kolkata, India, December 9-13, 2024, Proceedings, Part VIII. LNCS, vol. 15491, pp. 207–236. Springer (2024). https://doi.org/10.1007/978-981-96-0944-4_7
5. 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, ICLR 2023*, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023), <https://openreview.net/forum?id=-CoNloheTs>
6. Foerster, H., Mullins, R.D., Shumailov, I., Hayes, J.: Beyond slow signs in high-fidelity model extraction. In: Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J.M., Zhang, C. (eds.) *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, Vancouver, BC, Canada, December 10 - 15, 2024 (2024), http://papers.nips.cc/paper_files/paper/2024/hash/22ae669a35bb9e70eb93ab77c1eff5b4-Abstract-Conference.html
7. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G.J., Dunson, D.B., Dudík, M. (eds.) *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011*, Fort Lauderdale, USA, April 11-13, 2011. JMLR Proceedings, vol. 15, pp. 315–323. JMLR.org (2011), <http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf>
8. Ito, A., Miura, T., Todo, Y.: Is the hard-label cryptanalytic model extraction really polynomial? In: Heninger, N., Rosulek, M. (eds.) *Advances in Cryptology*

- CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part VII. Lecture Notes in Computer Science, vol. 16806, pp. 67–98. Springer (2026). https://doi.org/10.1007/978-3-032-35415-0_3, https://doi.org/10.1007/978-3-032-35415-0_3
- 9. Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., Papernot, N.: High accuracy and high fidelity extraction of neural networks. In: Capkun, S., Roesner, F. (eds.) 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020. pp. 1345–1362. USENIX Association (2020), <https://www.usenix.org/conference/usenixsecurity20/presentation/jagielski>
- 10. Liu, H., Siproudhis, A., Experton, S., Lorenz, P., Boura, C., Peyrin, T.: Navigating the deep: End-to-end extraction on deep neural networks. In: Daemen, J., Thomé, E. (eds.) Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part VI. pp. 482–512. Lecture Notes in Computer Science, Springer (2026). https://doi.org/10.1007/978-3-032-25333-0_17
- 11. Martinelli, F., Simsek, B., Gerstner, W., Brea, J.: Expand-and-cluster: Parameter recovery of neural networks. In: Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net (2024), <https://openreview.net/forum?id=3MIuPRJYwf>
- 12. Martinez, I.A.C., Chávez-Saab, J., Hambitzer, A., Rodríguez-Henríquez, F., Satspute, N., Shamir, A.: Polynomial time cryptanalytic extraction of neural network models. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part III. LNCS, vol. 14653, pp. 3–33. Springer (2024). https://doi.org/10.1007/978-3-031-58734-4_1
- 13. Milli, S., Schmidt, L., Dragan, A.D., Hardt, M.: Model reconstruction from model explanations. In: danah boyd, Morgenstern, J.H. (eds.) Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019. pp. 1–9. ACM (2019). <https://doi.org/10.1145/3287560.3287562>
- 14. Miura, T., Shibahara, T., Yanai, N.: MEGEX: data-free model extraction attack against gradient-based explainable AI. In: Proceedings of the 2nd ACM Workshop on Secure and Trustworthy Deep Learning Systems, SecTL 2024, Singapore, July 2, 2024. pp. 56–66. ACM (2024). <https://doi.org/10.1145/3665451.3665533>
- 15. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A.W., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. In: Black, A.W. (ed.) The 9th ISCA Speech Synthesis Workshop, SSW 2016, Sunnyvale, CA, USA, September 13-15, 2016. p. 125. ISCA (2016), https://www.isca-archive.org/ssw_2016/vandenoord16_ssw.html
- 16. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763. PMLR (2021), <http://proceedings.mlr.press/v139/radford21a.html>
- 17. Rolnick, D., Kording, K.P.: Reverse-engineering deep relu networks. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020,

- 13-18 July 2020, Virtual Event. Proceedings of Machine Learning Research, vol. 119, pp. 8178–8187. PMLR (2020), <http://proceedings.mlr.press/v119/rolnick20a.html>
18. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: Holz, T., Savage, S. (eds.) 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016. pp. 601–618. USENIX Association (2016), <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer>
 19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 5998–6008 (2017), <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>

A Recent Results Using an Artificial Model [2]

Very recently, the ePrint version of [2] was updated to include a black-box end-to-end hard-label model-extraction demonstration [1]. The authors constructed a toy model with 32 inputs, three hidden layers of width 32, and 10 outputs. They provide black-box procedures for collecting intersection points, grouping them, recovering signatures, and recovering signs. By combining these procedures, they demonstrate the feasibility of end-to-end model extraction.

However, this result should be interpreted carefully. Their target model is neither trained nor conventionally initialized. Its hidden-layer weights are orthogonal, and its biases are adjusted so that each hidden neuron is active for about half of the sampled inputs. Thus, it is a highly structured artificial model to demonstrate the feasibility of black-box hard-label model extraction.

This construction substantially reduces the difficulty of sign recovery. Their black-box implementation first attempts to find intersection points at which all neurons in the layers preceding the target layer are active. Once such an intersection point is found, the local linear map is full rank and isotropic in this model. Thanks to orthogonal weights, the off-side control direction is orthogonal to the off-side decision-boundary normal. Therefore, when the signatures and decision-boundary normals are recovered with sufficient precision, walking along the off side never changes the values in subsequent layers before a past toggle occurs. Consequently, no future toggle occurs on the off side. If a future toggle is detected before a past toggle, the corresponding direction can be identified as the on side.

Extending this technique beyond the artificial model is nontrivial. Without such a special artificial model, intersection points at which all preceding neurons are active are generally difficult to find reliably, whether the model is trained or conventionally initialized. The orthogonal weights, the activation-rate calibration, and the small number of hidden layers make such intersection points considerably easier to find. Even in the updated ePrint version [1], the experiments on a trained CIFAR-10 model remain white-box evaluations that use 10^2 – 10^3 intersection points. Thus, the feasibility of a black-box implementation for general non-artificial models remains to be verified.

B Collecting intersection spaces

In hard-label model extraction, the attacker first collects a sufficient number of intersection spaces. In this section, we describe how to collect the intersection spaces. An outline of this procedure is shown in Fig. 10.

Step 1: Find a point $\mathbf{x}_2$ on the decision boundary. We sample $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^{d_0}$ from an adequate distribution (e.g., a Gaussian distribution) such that $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$, and perform a binary search between them to obtain a boundary point $\mathbf{x}_2$.

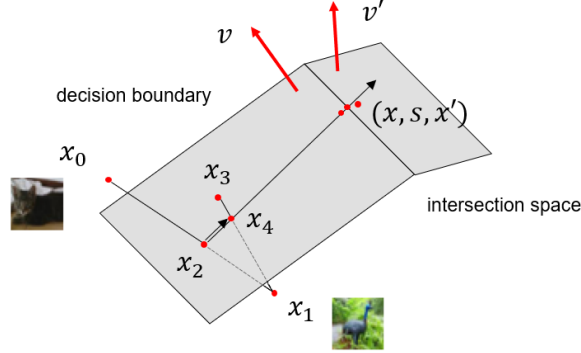


Fig. 10: Overview of how to collect intersection points and intersection spaces.

Step 2: Find another point $\mathbf{x}_4$ on the decision boundary. We move from $\mathbf{x}_2$ in a random direction to obtain $\mathbf{x}_3$ sufficiently close to the same decision boundary. Without loss of generality, assume $f(\mathbf{x}_3) = f(\mathbf{x}_0)$. A second boundary point $\mathbf{x}_4$ is obtained by a binary search between $\mathbf{x}_1$ and $\mathbf{x}_3$.

Step 3: Move along the decision boundary until reaching an intersection point $\mathbf{s}$. Let $\Delta\mathbf{x} := \mathbf{x}_4 - \mathbf{x}_2$ and $\mathbf{x}_\gamma := \mathbf{x}_2 + \gamma\Delta\mathbf{x}$ for a small γ . At this stage, we first recover the normal vector $\mathbf{v}$ of the local decision boundary and use it to guide the movement so that $\mathbf{x}_\gamma$ follows the decision boundary exactly along a straight line. As $\mathbf{x}_\gamma$ moves along the decision boundary, it eventually crosses an activation boundary of the ReLU network. The crossing point, called an intersection point, is denoted by $\mathbf{s}$. After crossing, the point is projected back onto the decision boundary to obtain $\mathbf{x}$, while the point immediately before crossing is denoted by $\mathbf{x}'$. This yields a triplet $(\mathbf{x}, \mathbf{s}, \mathbf{x}')$.

Step 4: Find the normal vectors $\mathbf{v}$ and $\mathbf{v}'$. For a boundary point $\mathbf{x} \in \mathbb{R}^{d_0}$ and a slight perturbation $\mathbf{x} + \alpha\mathbf{e}_1$, using the standard basis vectors $\mathbf{e}_1, \dots, \mathbf{e}_{d_0}$, we find β_i such that $\mathbf{x} + \alpha\mathbf{e}_1 + \beta_i\mathbf{e}_i$ lies on the decision boundary. The normal vector is aligned with $(1/\beta_1, \dots, 1/\beta_{d_0})^\top$ and normalized. Applying this procedure at $\mathbf{x}$ and $\mathbf{x}'$ gives the unit normals $\mathbf{v}$ and $\mathbf{v}'$.

Step 5: Define the intersection space. The dual space is the intersection of the two local decision-boundary hyperplanes determined by $\mathbf{v}$ and $\mathbf{v}'$, or equivalently, the orthogonal complement of their span. The intersection space $\mathcal{S}$ associated with $\mathbf{s}$ is defined as $\mathcal{S} := \mathbf{s} + \text{span}\{\mathbf{v}, \mathbf{v}'\}^\perp \subset \mathbb{R}^{d_0}$. We represent $\mathcal{S}$ by the triplet $(\mathbf{s}, \mathbf{v}, \mathbf{v}')$. In particular, let $\mathbf{n}_1, \dots, \mathbf{n}_{d_0-2}$ be an orthonormal basis of $\text{span}\{\mathbf{v}, \mathbf{v}'\}^\perp$. We then construct the basis matrix

$$\mathbf{N} := [\mathbf{n}_1 \cdots \mathbf{n}_{d_0-2}] \in \mathbb{R}^{d_0 \times (d_0-2)}.$$

Then, every point $\mathbf{x} \in \mathcal{S}$ can be parameterized as $\mathbf{x} = \mathbf{s} + \mathbf{N}\mathbf{u}$, $\mathbf{u} \in \mathbb{R}^{d_0-2}$.

Step 6: Repeat these steps. Repeating the procedure yields intersection spaces $\mathcal{S}_1, \dots, \mathcal{S}_N$, which are used in [2] to reconstruct the weight parameters.

C Pseudocode for Signature Recovery

The following algorithms formalize the consistency test and full signature recovery in Section 2.4. ISCONSISTENT tests for a shared affine constraint, which need not correspond to a target-layer neuron. RECOVERSIGNATURE returns a unit signature and its consistently scaled bias only when the full affine constraint has a one-dimensional normal space. In numerical implementations, ranks and null spaces require a tolerance; the formulas below describe exact arithmetic.

Algorithm 2 ISCONSISTENT

Require: $\mathcal{S}_1, \mathcal{S}_2$; recovered weights and biases through layer $\ell - 1$

Ensure: whether the pair passes the affine consistency test

- 1: **for** $i \in \{1, 2\}$ **do**
 - 2: Read $(\mathbf{s}_i, \mathbf{N}_i)$ from $\mathcal{S}_i$.
 - 3: $\mathbf{T}_i \leftarrow \mathbf{\Gamma}_{\mathbf{s}_i}^{(\ell-1)} \mathbf{N}_i$; $\mathbf{u}_i \leftarrow \mathbf{h}^{(\ell-1)}(\mathbf{s}_i)$.
 - 4: $\nu \leftarrow$ number of layer- $(\ell - 1)$ neurons active at either point.
 - 5: **if** $\ell = 1$ **then**
 - 6: $\nu \leftarrow d_0$.
 - 7: **return** $\text{rank}[\mathbf{T}_1, \mathbf{T}_2, \mathbf{u}_1 - \mathbf{u}_2] < \nu$.
-

Algorithm 3 RECOVERSIGNATURE

Require: $\mathcal{S}_1, \dots, \mathcal{S}_m$ for a candidate neuron in layer ℓ ; recovered preceding-layer weights and biases

Ensure: a unit signature $\mathbf{w}$ and bias b , or $\perp$ if not uniquely determined

- 1: **for** $i = 1, \dots, m$ **do**
 - 2: Read $(\mathbf{s}_i, \mathbf{N}_i)$ from $\mathcal{S}_i$.
 - 3: $\mathbf{T}_i \leftarrow \mathbf{\Gamma}_{\mathbf{s}_i}^{(\ell-1)} \mathbf{N}_i$; $\mathbf{u}_i \leftarrow \mathbf{h}^{(\ell-1)}(\mathbf{s}_i)$.
 - 4: $\mathbf{U} \leftarrow [\mathbf{T}_1, \dots, \mathbf{T}_m, \mathbf{u}_2 - \mathbf{u}_1, \dots, \mathbf{u}_m - \mathbf{u}_1]$.
 - 5: **if** $\text{rank}(\mathbf{U}) \neq d_{\ell-1} - 1$ **then**
 - 6: **return** $\perp$. $\triangleright$ Insufficient constraints or inconsistent data
 - 7: Choose a unit vector $\mathbf{w} \in \text{Ker}(\mathbf{U}^\top)$.
 - 8: $b \leftarrow -\mathbf{w}^\top \mathbf{u}_1$.
 - 9: **return** $(\mathbf{w}, b)$.
-

D On-Side Variance in Signature-Weighted Cosine

Let $\mathbf{A}$ be

$$\mathbf{A} = [\tilde{\mathbf{w}}_1, \dots, \tilde{\mathbf{w}}_{t-1}, \tilde{\mathbf{w}}_{t+1}, \dots, \tilde{\mathbf{w}}_{d_\ell}],$$

Then,

$$\tilde{\mathbf{C}} := \sum_{j \in \mathcal{J}_{\text{comp}}} (\alpha_j \tilde{\mathbf{w}}_j)(\alpha_j \tilde{\mathbf{w}}_j)^\top = \sum_{j \in \mathcal{J}_{\text{comp}}} \tilde{\mathbf{w}}_j \tilde{\mathbf{w}}_j^\top = \mathbf{A} \mathbf{A}^\top.$$

We assume $\text{rank}(\tilde{\mathbf{C}}) = r_\Gamma$ and interpret $\tilde{\mathbf{C}}^{-1}$ on $\text{Im}(\mathbf{P}_\Gamma)$. Then, we use the approximation

$$z_{\text{on}} \approx z_{\text{off}} + \frac{\alpha_t \beta_t \|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{w}}_t\|}{\|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{off}}\|}.$$

We first consider $\|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{off}}\|^2 = \tilde{\mathbf{v}}_{\text{off}}^\top \tilde{\mathbf{C}}^{-1} \tilde{\mathbf{v}}_{\text{off}}$. Let $\mathbf{a} \in \mathbb{R}^{|\mathcal{J}_{\text{comp}}|}$ be a zero-padded coefficient vector such that $\beta_j = a_j$ for $j \in \mathcal{J}_{\text{act}}$ and $a_j = 0$ otherwise. Since $\tilde{\mathbf{v}}_{\text{off}} = \mathbf{A} \mathbf{a}$,

$$\|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{off}}\|^2 = \mathbf{a}^\top (\mathbf{A}^\top (\mathbf{A} \mathbf{A}^\top)^{-1} \mathbf{A}) \mathbf{a} = \mathbf{a}^\top \mathbf{P}_\mathbf{A} \mathbf{a}.$$

Here, $\mathbf{P}_\mathbf{A}$ is an orthogonal projection matrix of rank r_Γ in the $|\mathcal{J}_{\text{comp}}|$ -dimensional coefficient space. Therefore, if the direction of $\mathbf{a}$ is not biased relative to the row space of $\mathbf{A}$,

$$\|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{off}}\|^2 \approx \frac{r_\Gamma}{|\mathcal{J}_{\text{comp}}|} \sum_{j \in \mathcal{J}_{\text{act}}} \beta_j^2.$$

Next, we consider $\|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{w}}_t\|^2 = \tilde{\mathbf{w}}_t^\top \tilde{\mathbf{C}}^{-1} \tilde{\mathbf{w}}_t$. If $\tilde{\mathbf{w}}_t$ were included among the generators of the competitor space, it would be expected to have the same value $r_\Gamma/|\mathcal{J}_{\text{comp}}|$ as the denominator. However, the actual $\tilde{\mathbf{C}}$ does not contain the target vector $\tilde{\mathbf{w}}_t$. If $\tilde{\mathbf{w}}_t$ has comparable components in each eigendirection of the competitor space, sparse directions are amplified and dense directions are weakened. For comparison, consider a reference model in which the target and competitors are independent Gaussian samples with the same covariance. By a property of the inverse of a Wishart matrix,

$$\mathbb{E} \left[\tilde{\mathbf{w}}_t^\top \tilde{\mathbf{C}}^{-1} \tilde{\mathbf{w}}_t \right] = \frac{r_\Gamma}{|\mathcal{J}_{\text{comp}}| - r_\Gamma - 1}.$$

In practice, however, the target of model extraction consists only of fixed parameters, and trained models may have biases in their weights. Therefore, this theoretical assumption does not generally hold. Instead, we measure the quantity directly at each intersection space and compare it. We used the same random and MNIST-trained models. Table 4 summarizes $\tilde{\mathbf{w}}_t^\top \tilde{\mathbf{C}}^{-1} \tilde{\mathbf{w}}_t$ for varying values of r_Γ . As a result, the estimate obtained from the above reference model remains a good approximation.

Table 4: Measured values of $\tilde{\mathbf{w}}_t^\top \tilde{\mathbf{C}}^{-1} \tilde{\mathbf{w}}_t$ and the reference-model prediction.

r_Γ	Random		MNIST		Prediction
	2nd layer	4th layer	2nd layer	4th layer	
50	0.6786	0.6602	0.6714	0.6699	0.6579
55	0.7629	0.7648	0.7500	0.7713	0.7746
60	0.9057	0.9049	0.9027	0.9012	0.9091
65	1.0550	1.0758	1.0501	1.0879	1.0656
70	1.2378	1.3441	1.2487	–	1.2500
75	1.4165	–	1.4326	–	1.4706

The typical squared ratio is estimated as

$$\left(\frac{\alpha_t \beta_t \|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{w}}_t\|}{\|\tilde{\mathbf{C}}^{-1/2} \tilde{\mathbf{v}}_{\text{off}}\|} \right)^2 \approx \frac{|\mathcal{J}_{\text{comp}}|}{|\mathcal{J}_{\text{comp}}| - r_\Gamma - 1} \frac{\beta_t^2}{\sum_{j \in \mathcal{J}_{\text{act}}} \beta_j^2}.$$

For the Γ -weighted cosine method, the on-side variance was estimated to exceed the off-side variance by $\frac{\beta_t^2}{\sum_{j \in \mathcal{J}_{\text{act}}} \beta_j^2} \approx \frac{1}{|\mathcal{J}_{\text{act}}|}$. For the signature-weighted cosine method, this amplification is multiplied by

$$\frac{|\mathcal{J}_{\text{comp}}|}{|\mathcal{J}_{\text{comp}}| - r_\Gamma - 1} \approx 2.$$

Consequently, under a normal approximation,

$$z_{\text{on}} \stackrel{\text{approx}}{\sim} \mathcal{N}\left(0, \frac{1}{r_\Gamma} + \frac{4}{d_\ell}\right).$$

E Derivation of Noise Components under Varying Activation Patterns

This section derives the mechanism described in case (ii) of Section 4: intersection spaces can pass the consistency check even when the j th neuron in layer ℓ has different activation states at their representative points. Suppose that each intersection space corresponds to the k th neuron in layer $\ell + 1$, and that the state of the j th neuron in layer ℓ is fixed in a neighborhood of each representative point. The class pair defining the decision boundary and the activation patterns from layer ℓ onward are shared, except for the j th neuron in layer ℓ and the k th neuron in layer $\ell + 1$. Activation patterns in layers preceding ℓ may vary freely. Under these conditions, intersection spaces whose representative points have different activation states for the j th neuron in layer ℓ can pass the consistency check together.

Let $\mathcal{J}$ be the index set of neurons in layer ℓ , excluding the j th neuron, that are active at all the points under consideration. The pre-activation of the k th

neuron in the next layer is

$$\hat{h}_k^{(\ell+1)} = \sum_{r \in \mathcal{J}} w_{k,r}^{(\ell+1)} h_r^{(\ell)} + b_k^{(\ell+1)} + w_{k,j}^{(\ell+1)} h_j^{(\ell)}$$

To describe the decision boundary, we work with logits before softmax. Following Section 2.1, consider an L -layer ReLU network whose final-layer output $\mathbf{h}^{(L)} = \mathbf{W}^{(L)} \mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}$ is the logit vector. Thus, the final layer considered here is a linear layer with a bias, without softmax. Applying softmax to the logits yields class probabilities, but preserves their ordering, including ties. Hence, the decision boundary of a classifier that selects the class with the largest logit can be described using logit differences, without applying softmax.

We derive the decision-boundary equation using the logit difference on the side where the k th neuron in layer $\ell + 1$ is inactive. We are concerned with intersections on this neuron's activation boundary, where $\hat{h}_k^{(\ell+1)} = 0$ and its output is zero. The expressions from the active and inactive sides therefore agree on the intersection, so the inactive-side expression suffices. Let $\mathbf{D}^{(\ell+1)}, \dots, \mathbf{D}^{(L-1)}$ be the downstream activation patterns on this side; in particular, the k th diagonal entry of $\mathbf{D}^{(\ell+1)}$ is zero. With these patterns fixed, the composition of the downstream layers is

$$\begin{aligned} \mathbf{h}^{(L)} &= \mathbf{B}^{(\ell)} \mathbf{h}^{(\ell)} + \mathbf{c}^{(\ell)}, \\ \mathbf{B}^{(\ell)} &= \mathbf{W}^{(L)} \mathbf{D}^{(L-1)} \mathbf{W}^{(L-1)} \dots \mathbf{D}^{(\ell+1)} \mathbf{W}^{(\ell+1)} \end{aligned}$$

Here, $\mathbf{c}^{(\ell)}$ is the constant vector obtained by composing the biases from layer $\ell + 1$ onward. Specifically, initialize $\mathbf{B}^{(L-1)} = \mathbf{W}^{(L)}$ and $\mathbf{c}^{(L-1)} = \mathbf{b}^{(L)}$, and compute, for $r = L - 1, \dots, \ell + 1$,

$$\begin{aligned} \mathbf{B}^{(r-1)} &= \mathbf{B}^{(r)} \mathbf{D}^{(r)} \mathbf{W}^{(r)}, \\ \mathbf{c}^{(r-1)} &= \mathbf{B}^{(r)} \mathbf{D}^{(r)} \mathbf{b}^{(r)} + \mathbf{c}^{(r)} \end{aligned}$$

Let c, c' be the classes defining the decision boundary. Their logit difference is

$$h_c^{(L)} - h_{c'}^{(L)} = (\mathbf{e}_c - \mathbf{e}_{c'})^\top \mathbf{B}^{(\ell)} \mathbf{h}^{(\ell)} + (\mathbf{e}_c - \mathbf{e}_{c'})^\top \mathbf{c}^{(\ell)}$$

where $\mathbf{e}_c$ and $\mathbf{e}_{c'}$ are the corresponding standard basis vectors. Define

$$\boldsymbol{\beta} = \mathbf{B}^{(\ell)\top} (\mathbf{e}_c - \mathbf{e}_{c'}), \quad b_{\text{down}} = (\mathbf{e}_c - \mathbf{e}_{c'})^\top \mathbf{c}^{(\ell)}$$

Then β_r is the coefficient describing the contribution of $h_r^{(\ell)}$ to the logit difference, and b_{down} accounts for the downstream biases. The decision-boundary equation is therefore

$$\sum_{r \in \mathcal{J}} \beta_r h_r^{(\ell)} + b_{\text{down}} + \beta_j h_j^{(\ell)} = 0$$

In particular, β_j is the coefficient of the output $h_j^{(\ell)}$ of the j th neuron in layer ℓ and is shared by both activation states.

Substitute $h_r^{(\ell)} = \mathbf{w}_r^{(\ell)\top} \mathbf{h}^{(\ell-1)} + b_r^{(\ell)}$ for each $r \in \mathcal{J}$, and collect the contributions of all neurons other than the j th neuron by defining

$$\begin{aligned}\tilde{\mathbf{w}}_k^{(\ell+1)} &= \sum_{r \in \mathcal{J}} w_{k,r}^{(\ell+1)} \mathbf{w}_r^{(\ell)}, & \tilde{b}_k^{(\ell+1)} &= \sum_{r \in \mathcal{J}} w_{k,r}^{(\ell+1)} b_r^{(\ell)} + b_k^{(\ell+1)}, \\ \mathbf{v}^{(\ell)} &= \sum_{r \in \mathcal{J}} \beta_r \mathbf{w}_r^{(\ell)}, & b_{\text{DB}}^{(\ell)} &= \sum_{r \in \mathcal{J}} \beta_r b_r^{(\ell)} + b_{\text{down}}\end{aligned}$$

When the j th neuron in layer ℓ is inactive, $h_j^{(\ell)} = 0$, so the intersection satisfies

$$\begin{aligned}\tilde{\mathbf{w}}_k^{(\ell+1)\top} \mathbf{h}^{(\ell-1)} + \tilde{b}_k^{(\ell+1)} &= 0, \\ \mathbf{v}^{(\ell)\top} \mathbf{h}^{(\ell-1)} + b_{\text{DB}}^{(\ell)} &= 0\end{aligned}$$

When this neuron is active, $h_j^{(\ell)} = \hat{h}_j^{(\ell)} = \mathbf{w}_j^{(\ell)\top} \mathbf{h}^{(\ell-1)} + b_j^{(\ell)}$. Substituting this expression yields

$$\begin{aligned}\tilde{\mathbf{w}}_k^{(\ell+1)\top} \mathbf{h}^{(\ell-1)} + \tilde{b}_k^{(\ell+1)} + w_{k,j}^{(\ell+1)} \hat{h}_j^{(\ell)} &= 0, \\ \mathbf{v}^{(\ell)\top} \mathbf{h}^{(\ell-1)} + b_{\text{DB}}^{(\ell)} + \beta_j \hat{h}_j^{(\ell)} &= 0\end{aligned}$$

Thus, in the input space of layer ℓ , the activation-boundary normal and the decision-boundary normal acquire the additional terms $w_{k,j}^{(\ell+1)} \mathbf{w}_j^{(\ell)}$ and $\beta_j \mathbf{w}_j^{(\ell)}$, respectively. However, multiplying the activation-boundary equation by β_j and subtracting $w_{k,j}^{(\ell+1)}$ times the decision-boundary equation eliminates this neuron's contribution. For both activation states, we obtain

$$(\beta_j \tilde{\mathbf{w}}_k^{(\ell+1)} - w_{k,j}^{(\ell+1)} \mathbf{v}^{(\ell)})^\top \mathbf{h}^{(\ell-1)} + \beta_j \tilde{b}_k^{(\ell+1)} - w_{k,j}^{(\ell+1)} b_{\text{DB}}^{(\ell)} = 0$$

If this normal is nonzero, the intersection spaces for both activation states lie in a common hyperplane in the input space of layer ℓ and can therefore pass the consistency check. Thus, even when both boundaries bend as the j th neuron in layer ℓ switches state, a shared linear constraint on their intersections remains. Even if activation patterns in preceding layers differ, the tangent directions of each intersection space, mapped to the input space of layer ℓ by the corresponding $\mathbf{\Gamma}_{\mathbf{s}_i}^{(\ell-1)}$, are orthogonal to this common normal. The activation patterns in preceding layers therefore need not be fixed.

F Cross-Layer Extraction of Unreachable Weights

We describe how cross-layer extraction [8] applies to unreachable weights in the setting of Section 4. If a neuron with an unreachable weight is active at intersection points of a neuron in the next layer, the corresponding intersection spaces can be used for cross-layer recovery. Let k be the index of the target neuron in layer ℓ , and suppose that its signature has been recovered except for coordinate j . Consider the η th neuron in layer $\ell + 1$, whose activation boundary

satisfies $\mathbf{w}_\eta^{(\ell+1)\top} \mathbf{h}^{(\ell)} + b_\eta^{(\ell+1)} = 0$. Let $\mathcal{P} = [d_\ell] \setminus \{k\}$ be the index set of the other neurons, which have no unreachable weights in the setting considered here. Let $\mathbf{w}_{\eta,\mathcal{P}}^{(\ell+1)}$ and $\mathbf{h}_\mathcal{P}^{(\ell)}$ be the restrictions of the next-layer weight vector and the layer- ℓ activation vector to $\mathcal{P}$, respectively. The boundary equation becomes

$$\mathbf{w}_{\eta,\mathcal{P}}^{(\ell+1)\top} \mathbf{h}_\mathcal{P}^{(\ell)} + w_{\eta,k}^{(\ell+1)} h_k^{(\ell)} + b_\eta^{(\ell+1)} = 0.$$

When the target neuron is active, $h_k^{(\ell)} = \mathbf{w}_k^{(\ell)\top} \mathbf{h}^{(\ell-1)} + b_k^{(\ell)}$. Substituting this expression gives

$$\mathbf{w}_{\eta,\mathcal{P}}^{(\ell+1)\top} \mathbf{h}_\mathcal{P}^{(\ell)} + w_{\eta,k}^{(\ell+1)} \mathbf{w}_k^{(\ell)\top} \mathbf{h}^{(\ell-1)} + w_{\eta,k}^{(\ell+1)} b_k^{(\ell)} + b_\eta^{(\ell+1)} = 0.$$

Equivalently,

$$(\mathbf{w}_{\eta,\mathcal{P}}^{(\ell+1)\top}, w_{\eta,k}^{(\ell+1)} \mathbf{w}_k^{(\ell)\top}) \begin{pmatrix} \mathbf{h}_\mathcal{P}^{(\ell)} \\ \mathbf{h}^{(\ell-1)} \end{pmatrix} + w_{\eta,k}^{(\ell+1)} b_k^{(\ell)} + b_\eta^{(\ell+1)} = 0.$$

Thus, in the augmented space formed by concatenating the inputs to layers $\ell + 1$ and ℓ as above, this equation defines an effective activation boundary. Its normal is $(\mathbf{w}_{\eta,\mathcal{P}}^{(\ell+1)\top}, w_{\eta,k}^{(\ell+1)} \mathbf{w}_k^{(\ell)\top})^\top$, and its bias is $w_{\eta,k}^{(\ell+1)} b_k^{(\ell)} + b_\eta^{(\ell+1)}$. Applying the signature-recovery procedure for hard-label extraction to this boundary recovers the target signature multiplied by the next-layer weight. Unlike the original cross-layer setting, all coordinates of $\mathbf{w}_k^{(\ell)}$ except coordinate j have already been recovered. These known coordinates allow us to determine $w_{\eta,k}^{(\ell+1)}$ and then recover the missing coordinate $w_{k,j}^{(\ell)}$. The same approach extends to unreachable weights in multiple neurons within a layer.

G Intersection-Point Search Procedures

G.1 Searching for Informative Intersection Points

Problem setting. We describe the search for informative intersection points used to recover missing signature coordinates in Section 4. Consider the k th neuron in layer ℓ . Let $\{(\mathbf{s}_i, \mathbf{v}_i, \mathbf{v}'_i)\}_{i \in [m]}$ be the representations of the collected intersection spaces for this neuron, where $\mathbf{s}_i$ is a representative intersection point and $\mathbf{v}_i, \mathbf{v}'_i$ are the adjacent decision-boundary normals. Suppose that the j th neuron in layer $\ell - 1$ is inactive at every collected point. Thus, the j th input coordinate to layer ℓ is zero at $\mathbf{s}_1, \dots, \mathbf{s}_m$. Let $\Lambda = [d_{\ell-1}] \setminus \{j\}$ be the index set excluding the missing coordinate, and let $\mathbf{w}_{k,\Lambda}^{(\ell)}$ and $\mathbf{h}_\Lambda^{(\ell-1)}$ be the restrictions of the target signature and its input vector to Λ , respectively. The target neuron's pre-activation is

$$\hat{h}_k^{(\ell)} = \mathbf{w}_{k,\Lambda}^{(\ell)\top} \mathbf{h}_\Lambda^{(\ell-1)} + b_k^{(\ell)} + w_{k,j}^{(\ell)} h_j^{(\ell-1)}.$$

Here, $w_{k,j}^{(\ell)}$ is the j th coordinate of the target weight vector, and $h_j^{(\ell-1)}$ is the corresponding input coordinate. At each collected intersection point, $h_j^{(\ell-1)} = 0$

and the target neuron lies on its activation boundary, so

$$\mathbf{w}_{k,A}^{(\ell)\top} \mathbf{h}_A^{(\ell-1)} + b_k^{(\ell)} = 0.$$

Let $\hat{h}_{\text{part}}^{(\ell)}$ be the partial pre-activation obtained by omitting coordinate j :

$$\hat{h}_{\text{part}}^{(\ell)}(\mathbf{x}) = \mathbf{w}_{k,A}^{(\ell)\top} \mathbf{h}_A^{(\ell-1)}(\mathbf{x}) + b_k^{(\ell)}.$$

Before sign recovery, the orientation of the recovered activation boundary remains ambiguous.⁷

Searching for informative intersection points. The first approach searches for an intersection point at which $h_j^{(\ell-1)} > 0$. Choose an existing intersection point of the target neuron, say $\mathbf{s}_1$, and search along the intersection for a point $\mathbf{x}_0$ satisfying $\hat{h}_j^{(\ell-1)}(\mathbf{x}_0) = 0$. By definition, $\mathbf{s}_1$ lies on the model’s decision boundary and satisfies $\hat{h}_k^{(\ell)}(\mathbf{s}_1) = 0$. Since the weights and signs in preceding layers have already been recovered, we can evaluate $\hat{h}_j^{(\ell-1)}$. The preceding neuron is inactive at the starting point, with $\hat{h}_j^{(\ell-1)}(\mathbf{s}_1) < 0$.

We move tangentially to both the activation boundary and the decision boundary, maintaining $\hat{h}_{\text{part}}^{(\ell)}(\mathbf{x}) = 0$ while remaining on the decision boundary, until $\hat{h}_j^{(\ell-1)}$ reaches zero. To choose a direction, we project the ascent direction $\nabla_{\mathbf{x}} \hat{h}_j^{(\ell-1)}(\mathbf{x})$ onto the orthogonal complement of the span of two normals: the activation-boundary normal $\nabla_{\mathbf{x}} \hat{h}_{\text{part}}^{(\ell)}(\mathbf{x})$ and either adjacent decision-boundary normal, $\mathbf{v}_1$ or $\mathbf{v}'_1$. If the search crosses a recovered activation boundary in a preceding layer, the local linear map changes, and we update the direction accordingly. Crossing an activation boundary in a subsequent layer can also invalidate the direction, but detecting such a crossing is difficult because that layer has not yet been recovered. If the search fails because of such a crossing, we restart from another intersection point of the target neuron.

Once a point $\mathbf{x}_0$ satisfying $\hat{h}_j^{(\ell-1)}(\mathbf{x}_0) = 0$ is found on the target intersection, we perturb it to obtain a point $\mathbf{x}_1$ with $\hat{h}_j^{(\ell-1)}(\mathbf{x}_1) = \epsilon$, where ϵ is a sufficiently small positive value. The perturbed point remains close to both boundaries, while the input coordinate corresponding to the missing weight is now nonzero. Starting from $\mathbf{x}_1$, we search for an intersection point using the procedure described in Section B. This may yield a point that reveals the missing coordinate. If the attempt fails, we restart from another intersection point associated with the target neuron. In our experiments, we try at most 16 starting intersection points.

⁷ The signs of $\mathbf{w}_k^{(\ell)}$ and $b_k^{(\ell)}$ have not yet been recovered. This sign ambiguity does not affect the search along the activation boundary described below, which does not require distinguishing its active and inactive sides.

G.2 Searching for Intersections to Validate Candidate Signatures

We describe the intersection search used to validate candidate signatures in Section 4. Suppose that the signatures and signs in all layers preceding layer ℓ have been recovered. Let $\mathbf{w}$ and b be the candidate signature and its bias, respectively. The candidate pre-activation is $\hat{h}(\mathbf{x}) = \mathbf{w}^\top \mathbf{h}^{(\ell-1)}(\mathbf{x}) + b$.

We reuse the intersection points already collected for signature recovery. Let $\mathbf{s}_1, \dots, \mathbf{s}_m$ be these points after excluding those used to recover the candidate signature, since reusing the latter would not provide an independent check. Each intersection point is adjacent to two local pieces of the decision boundary. For each $\mathbf{s}_i$, let $\mathbf{x}_i$ and $\mathbf{x}'_i$ be points on these two pieces, and let $\mathbf{v}_i$ and $\mathbf{v}'_i$ be their respective normals. Choose one of the pieces, say the one containing $\mathbf{x}_i$ with normal $\mathbf{v}_i$. Let $\mathbf{\Gamma}_i^{(\ell-1)}$ be the forward local linear matrix through layer $\ell - 1$ at $\mathbf{x}_i$. The input-space normal of the candidate activation boundary is $\nabla_{\mathbf{x}} \hat{h}(\mathbf{x}_i) = \mathbf{\Gamma}_i^{(\ell-1)\top} \mathbf{w}$. We seek the intersection by moving toward the candidate activation boundary while remaining on the chosen decision-boundary piece.

The pre-activation gradient projected onto the tangent space of the decision boundary is

$$\nabla_{\parallel \text{DB}} \hat{h}(\mathbf{x}_i) = \mathbf{\Gamma}_i^{(\ell-1)\top} \mathbf{w} - \frac{\mathbf{v}_i (\mathbf{v}_i^\top \mathbf{\Gamma}_i^{(\ell-1)\top} \mathbf{w})}{\|\mathbf{v}_i\|_2^2}.$$

When this projected gradient is nonzero, moving against it if $\hat{h}(\mathbf{x}_i) > 0$, or along it if $\hat{h}(\mathbf{x}_i) < 0$, brings us closer to the candidate boundary. If the intersection can be reached without crossing another activation boundary, the required displacement is

$$\Delta \mathbf{x}_i = -\frac{\hat{h}(\mathbf{x}_i)}{\|\nabla_{\parallel \text{DB}} \hat{h}(\mathbf{x}_i)\|_2} \frac{\nabla_{\parallel \text{DB}} \hat{h}(\mathbf{x}_i)}{\|\nabla_{\parallel \text{DB}} \hat{h}(\mathbf{x}_i)\|_2}.$$

The corresponding distance is $|\hat{h}(\mathbf{x}_i)| / \|\nabla_{\parallel \text{DB}} \hat{h}(\mathbf{x}_i)\|_2$. Under this condition, $\mathbf{x}_i + \Delta \mathbf{x}_i$ lies at the intersection of the decision boundary and the candidate activation boundary. We then check for a bend in the decision boundary to assess the candidate signature.

In practice, we compute the predicted distance $|\hat{h}(\mathbf{x}_i)| / \|\nabla_{\parallel \text{DB}} \hat{h}(\mathbf{x}_i)\|_2$ for each starting point $\mathbf{x}_i$ and attempt the searches in ascending order of this distance. Prioritizing nearby intersections reduces the likelihood of crossing other ReLU cells. Empirically, this procedure reaches the desired intersection in most cases.

H Sign Correction Using Decision-Boundary Normals

We detail Step 2 of the sign-recovery procedure in Section 4. The signs assigned in Step 1 are not necessarily correct because numerical errors and incomplete recovery can affect the underlying sign estimates. In this stage, we check their

consistency with the decision-boundary normals already collected and revise the assignments to improve that consistency. The key property is that each decision-boundary normal lies in the span of the signatures of neurons active on its corresponding side, after these signatures are pulled back to the input space.

Let m be the number of intersection spaces collected for signature recovery in the same layer. The representation of each space includes one normal from each of the two decision-boundary pieces adjacent to its representative point, giving $2m$ normals in total. Let $\mathbf{v}_1, \dots, \mathbf{v}_{2m}$ be these normals. For each normal, we use the activation states at the nearby point on the side from which it was obtained.

Let $\mathcal{J}_{\text{rec}}$ be the index set of recovered neurons in the layer, and let $\boldsymbol{\alpha}$ be their current sign assignment. Let $\mathcal{J}_{\text{inactive},i}(\boldsymbol{\alpha})$ be the subset classified as inactive under $\boldsymbol{\alpha}$ at the nearby point where $\mathbf{v}_i$ was obtained. Following the candidate-set construction in Section 3, define $\mathcal{J}_{\text{cand},i}(\boldsymbol{\alpha}) = \mathcal{J}_{\text{rec}} \setminus \mathcal{J}_{\text{inactive},i}(\boldsymbol{\alpha})$. Here every recovered neuron has an assigned sign, so this candidate set consists of neurons classified as active. Unlike $\mathcal{J}_{\text{comp}}$ in single-neuron sign recovery, $\mathcal{J}_{\text{rec}}$ excludes no target neuron: each normal is checked using all recovered neurons classified as active on its side. Let $\mathbf{\Gamma}_i$ be the forward local linear matrix to the target layer’s input at that point. The signature $\mathbf{w}_j$ pulls back to the input-space normal $\mathbf{\Gamma}_i^\top \mathbf{w}_j$. Since the observed normal $\mathbf{v}_i$ is also an input-space vector, we compare it with the span $\mathcal{V}_i(\boldsymbol{\alpha}) = \text{span}\{\mathbf{\Gamma}_i^\top \mathbf{w}_j \mid j \in \mathcal{J}_{\text{cand},i}(\boldsymbol{\alpha})\}$. Flipping a sign changes not only the orientation of a signature but also which neurons are classified as active and hence included in this span. If all contributing neurons have been recovered accurately and their signs are correct, each normal lies in its corresponding span.

We define the *violation count* as the number of normals that lie outside their corresponding spans: $V(\boldsymbol{\alpha}) = \#\{i \in [2m] \mid \mathbf{v}_i \notin \mathcal{V}_i(\boldsymbol{\alpha})\}$. We use this count to update the signs as follows.

1. **Evaluate the current assignment.** Initialize the signs using the result of Step 1. Construct $\mathcal{J}_{\text{cand},i}(\boldsymbol{\alpha})$ and its corresponding span for each normal, and compute $V(\boldsymbol{\alpha})$.
2. **Evaluate individual sign flips.** For each recovered neuron, consider the assignment obtained by flipping only its sign. Update $\mathcal{J}_{\text{inactive},i}(\boldsymbol{\alpha})$ and $\mathcal{J}_{\text{cand},i}(\boldsymbol{\alpha})$ and recompute the violation count for each candidate assignment.
3. **Update or terminate.** If any candidate has a lower violation count than the current assignment, flip the sign that yields the largest reduction. Repeat the evaluation using the updated assignment. Terminate when no single sign flip reduces the violation count.

Contributions from unrecovered almost-dead neurons and numerical errors may prevent the violation count from reaching zero even when the signs are correct. Nevertheless, an appropriate sign assignment is expected to restore consistency with many of the observed normals and yield a small violation count.