Beyond Feedforward Networks: Cryptanalytic Extraction of RNNs

Longxiang Wei[1,2,3], Hao Lei[1,2,3], Xiaokang Qi[1,2,3], Xiaohan Sun[1,2,3], Lei Gao[1], Kai Hu[1,3,4,5], Wei Wang[1,2,3,4], and Meiqin Wang[1,2,3] (✉)

[1] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
longxiangwei@mail.sdu.edu.cn,leihao@mail.sdu.edu.cn,xiaokangqi@mail.sdu.edu.cn, xhansun@mail.sdu.edu.cn,leigao@sdu.edu.cn,kai.hu@sdu.edu.cn, weiwangsdu@sdu.edu.cn,mqwang@sdu.edu.cn
[2] State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao, 266237, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China.
[4] Quan Cheng Shandong Laboratory, Jinan, China
[5] Suzhou Research Institute, Shandong University, Suzhou, 215123, China

Abstract. Recurrent neural networks (RNNs) play an important role in time series modeling, signal processing, and resource constrained applications. Their parameters encode valuable functionality and constitute proprietary intellectual property. Cryptanalytic extraction tests whether black box queries reveal functionally equivalent parameters. Such analysis is important for assessing the security of deployed RNNs. However, prior cryptanalytic attacks mainly target feedforward networks such as fully connected networks (FCNs) and convolutional neural networks (CNNs). RNNs reuse their recurrent parameters at every time step, which existing extraction pipelines do not address.

We present cryptanalytic extraction of ReLU RNNs, including single-layer and stacked (a.k.a. multi-layer) RNNs, in both the hard-label and raw-output settings. To the best of our knowledge, this is the first such extraction. Using controlled inputs, we construct equivalent feedforward models whose depths depend on the number of recurrent layers, independently of the sequence length. However, parameter recovery from the equivalent models may yield recurrent weights with incompatible neuron orderings and positive scalings for the previous and current hidden states. We develop *recurrent parameter alignment* to make these transformations consistent and enable parameter reuse across time. Stacked RNNs are harder, because one equivalent model contains coupled branches, which prevent direct use of FCN sign recovery and complicate signature recovery. We therefore further develop two techniques, *recurrent parameter recovery* and *folded bias difference*. The former uses the recovered input weights to guide clustering and recover the remaining recurrent parameters, including their signs. The latter handles neurons that are otherwise difficult to recover.

We demonstrate end-to-end extraction across five RNN architectures with up to three recurrent layers in both observation settings. An ad-

ditional raw-output experiment extends the sequence length from 20 to 1024 using the same target parameters.

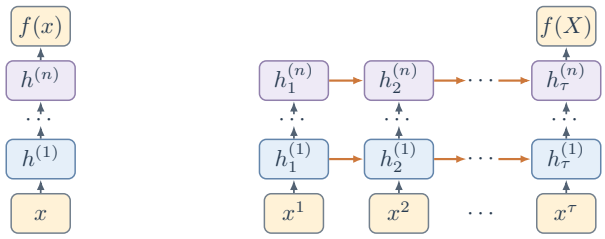
Keywords: ReLU Neural Networks · Functionally Equivalent Extraction · Recurrent Neural Networks · Hard Label.

1 Introduction

Recurrent neural networks (RNNs) support sequence modeling and signal processing, including applications with latency and resource constraints such as calorimeter energy reconstruction on FPGAs [1] and human activity recognition [20]. Their trained parameters embody substantial development investment and constitute valuable intellectual property, which motivates the study of parameter extraction attacks. Cryptanalytic extraction [4,6,7,8] seeks to recover a functionally equivalent model through black-box queries. For ReLU networks, the recovered parameters may differ from the target in neuron ordering and positive scaling, and two observation settings are distinguished by what the queries return. The raw-output setting exposes the real-valued logit, whereas the hard-label setting exposes only the predicted label, and in both settings target logits and target parameters are unavailable during attack recovery.

Prior cryptanalytic extraction has primarily targeted fully connected networks (FCNs) and convolutional neural networks (CNNs), whose layers carry separate parameter sets. RNNs instead reuse their weights and biases across time [13], so the recovered parameters must remain consistent across recurrent state updates, and this temporal parameter sharing raises challenges beyond those addressed by existing FCN extraction pipelines. This paper studies cryptanalytic extraction of ReLU RNNs under this requirement. The targets are unidirectional many-to-one RNNs with either a single recurrent layer or several stacked ones. We consider ungated ReLU recurrence [21] and exclude gated architectures such as LSTM [15] and GRU [11]. The attacker controls every real-valued input vector in the sequence but observes only the final output.

Cryptanalytic extraction of ReLU FCNs has matured in both settings. In the raw-output setting, Jagielski et al. [17] made the extraction of single-hidden-layer ReLU networks practical with finite-difference gradient estimation, and the geometric approach of Rolnick and Kording [28] extended extraction to deeper ReLU networks in theory, although its practical implementation remained limited to the first two hidden layers and incurred high query costs [28,7]. Carlini et al. [7] then demonstrated (ε, δ)-functionally equivalent extraction of FCNs with up to three hidden layers by layerwise differential extraction, and Canales-Martínez et al. [4] made sign recovery polynomial time with the SOE, Neuron Wiggle, and last-layer methods. Foerster et al. [14] accelerated sign recovery further, and Liu et al. [23] extracted FCNs with eight hidden layers end-to-end, that is, recovering all weight matrices and bias vectors needed to run the model, using only black-box queries. In the hard-label setting, Chen et al. [8] gave the first parameter extraction attack on FCNs, Carlini et al. [6] obtained polynomial-time



(a) FCN **(b) Unfolded stacked RNN**

Fig. 1: Structural comparison between an FCN with n hidden layers and a stacked RNN with n recurrent layers unfolded over time. FCN layers use separate parameter sets, whereas each RNN layer reuses its parameters across time, and only the final state of the top recurrent layer feeds the output.

signature and sign recovery from dual points, Canales-Martínez and Santos [5] recovered multiclass output layers and adapted SOE to hard-label sign recovery, and Ito et al. [16] exploited cross-layer interactions to recover persistently active neurons.

Beyond ReLU FCNs, these methods have been extended to other activation functions, namely PReLU [9,27,24], HardTanh [27], GELU [2], and sigmoid [2], and to other architectures. CNNs have been extracted with physical side channels [12], with average pooling under hard-label access [29], and with max pooling [10,22]. For Transformers, Wang et al. [32] extract multihead softmax attention models. This line of work has not reached RNNs. Prior RNN extraction work by Takemura et al. [30] used outputs at intermediate time steps to train substitute models, whereas our setting exposes only the final output.

Fig. 1 contrasts an FCN, whose n hidden layers carry separate parameter sets, with an RNN unfolded over τ time steps, which reuses the parameters of each of its n recurrent layers at every step. Since the unfolded RNN is itself a feedforward graph, one might hope to apply FCN extraction to it directly, but three obstacles prevent this.

Challenge 1: Depth growth under temporal unfolding. Temporal unfolding produces a feedforward graph whose depth grows with the sequence length, even for a single-layer RNN, so direct layerwise extraction would have to traverse many nonlinear transformations. The challenge is to construct shallow models whose depth is independent of the sequence length but which retain the information needed to recover the recurrent parameters.

Challenge 2: Recurrent consistency. FCN extraction [6,7,8] recovers each layer only up to the ordering and positive scaling of its neurons. In an FCN, adjacent layers compensate for these transformations, whereas recurrent parameters are applied at every time step, so a mismatch between the transformations recovered on their input and output sides does not cancel across time steps. The challenge

is to align neuron ordering and scaling so that the recovered recurrent parameters can be reused across time. We call this requirement *recurrent consistency* and formalize it in Definition 6.

Challenge 3: Branch interference after temporal unfolding. For stacked RNNs, the equivalent model that exposes the recurrent parameters contains two coupled branches, which represent the states at the final two time steps and are connected through recurrent weights, and this coupling interferes with both stages of FCN extraction. Sign recovery is affected first, because an input perturbation that targets one neuron can also follow paths through the other branch that bypass the neuron and change the output even when it is inactive, so the output response no longer isolates its activation state. Signature recovery is affected as well, because in deeper recurrent layers each neuron combines input weights that are already known with recurrent weights that are not, yet conventional signature recovery treats both blocks as unknown, which complicates the clustering of critical or dual points by neuron. The challenge is to use the known block to guide clustering and parameter recovery while keeping the recovered signs, ordering, and scaling consistent.

Contributions. We extract single-layer and stacked ReLU RNNs in both observation settings, which to the best of our knowledge is the first such cryptanalytic extraction, and our three contributions address the three challenges in turn.

- **Reduction to feedforward model recovery.** In Sections 3.1 and 4.1, we vary only the final or the penultimate input and fix all other inputs to zero. For an RNN with n recurrent layers, this yields two equivalent feedforward models, the final-input model $M_1^{(n)}$ and the penultimate-input model $M_2^{(n)}$, whose depths are determined by n rather than by the sequence length, because the fixed input prefix contributes only constant bias terms. Existing FCN techniques recover the input weights and the output parameters from $M_1^{(n)}$, after which $M_2^{(n)}$ exposes the recurrent parameters.
- **Recovering recurrent consistency.** In Section 3.2, we develop Recurrent Parameter Alignment (RPA) to enforce recurrent consistency. RPA matches folded biases, the effective bias terms obtained by fixing the earlier inputs, as recovered from queries against those computed from the recovered parameters, and these matches determine the neuron ordering and scaling. Theorem 1 shows that, under exact recovery, the aligned RNN agrees with the target on every input sequence up to the output scale that is unavoidable in the hard-label setting. In Section 4.2, we extend RPA to stacked RNNs, where it also resolves the signs of the first recurrent layer.
- **Parameter recovery with coupled branches.** In Sections 4.3 and 4.4, we develop Recurrent Parameter Recovery (RPR) and Folded Bias Difference (FBD), which recover the remaining recurrent parameters in $M_2^{(n)}$. In each subsequent layer, RPR takes the input weight rows already recovered from $M_1^{(n)}$ as neuron anchors, which fix part of the critical hyperplane normal and

Table 1: Model extraction results. Our RNN evaluation uses $|\mathcal{S}| = 10^5$ complete input sequences. Their coordinates are sampled independently from $\mathcal{N}(0, 1)$. The absolute and relative metrics follow Definition 3. $\triangle$ denotes the maximum absolute output error. $\diamond$ denotes the 75th-percentile relative output error. The query count for [23] sums the signature and sign recovery counts. For FCN rows, "Depth" is the number of hidden layers. For RNN rows, "Depth" counts recurrent layers. "Time Steps" denotes the sequence length. ✓ indicates an end-to-end attack. §: Data from Section 8 of the extended version of [6] have per-layer relative parameter errors of at most 4×10^{-14}; end-to-end output errors are not reported.

Architecture	Type	Depth	Time Steps	End-to-End	Setting	$(\varepsilon, 0)^{\triangle}$	$(\varepsilon, 0.25)^{\diamond}$	Queries	Reference
40-20-10^2-1	FCN	3	–	✓	Raw-output	$2^{-23.4}$	–	$2^{17.8}$	[7]
3072-256^8-10	FCN	8	–	✗	Raw-output	–	–	–	[4]
784-8^8-1	FCN	8	–	✓	Raw-output	–	$2^{-15.04}$	$2^{28.14}$	[23]
200-32-1	RNN	1	20	✓	Raw-output	$2^{-23.31}$	$2^{-26.65}$	$2^{22.07}$	This work
200-32-1	RNN	1	1024	✓	Raw-output	$2^{-23.30}$	$2^{-26.83}$	$2^{21.58}$	This work
200-32-24-1	RNN	2	20	✓	Raw-output	$2^{-20.94}$	$2^{-24.72}$	$2^{23.06}$	This work
200-32^2-1	RNN	2	20	✓	Raw-output	$2^{-20.89}$	$2^{-24.88}$	$2^{24.03}$	This work
200-32-24-16-1	RNN	3	20	✓	Raw-output	$2^{-17.25}$	$2^{-20.45}$	$2^{24.32}$	This work
200-32^3-1	RNN	3	20	✓	Raw-output	$2^{-12.19}$	$2^{-14.64}$	$2^{22.78}$	This work
1024-2^2-1	FCN	2	–	✓	Hard-label	$2^{-14.16}$	–	$2^{22.49}$	[8]
32-32^3-10	FCN	3	–	✓	Hard-label	–	–	$2^{25.24}$	[6]§
200-32-1	RNN	1	20	✓	Hard-label	$2^{-18.46}$	$2^{-21.03}$	$2^{26.48}$	This work
200-32-24-1	RNN	2	20	✓	Hard-label	$2^{-12.34}$	$2^{-15.88}$	$2^{32.14}$	This work
200-32^2-1	RNN	2	20	✓	Hard-label	$2^{-13.08}$	$2^{-17.25}$	$2^{32.32}$	This work
200-32-24-16-1	RNN	3	20	✓	Hard-label	$2^{-12.07}$	$2^{-16.00}$	$2^{31.69}$	This work
200-32^3-1	RNN	3	20	✓	Hard-label	$2^{-11.11}$	$2^{-13.11}$	$2^{32.51}$	This work

guide the clustering of critical or dual points, so that only the recurrent weight rows and biases remain to be recovered, with consistent neuron ordering and scaling and without a separate sign recovery stage. FBD complements RPR for difficult neurons by obtaining folded biases from shallower probe models, whose differences eliminate the unknown neuron bias and leave equations for the recurrent weights. Theorem 2 establishes functional equivalence of the recovered stacked RNN under the same exact recovery assumptions.

In Section 5, we evaluate five RNN architectures with one to three recurrent layers in both observation settings, the largest being 200-32^3-1, and we additionally rerun raw-output extraction of the 200-32-1 target at $\tau = 1024$ with the same target parameters as the $\tau = 20$ baseline. Table 1 reports the extraction results.

Outline. Section 2 gives preliminaries and the attack model, Sections 3 and 4 extract single-layer and stacked RNNs, Section 5 reports experiments, and Section 6 discusses limitations and defenses.

2 Preliminaries

2.1 Notation and Definitions

This section reviews notation and definitions for FCN extraction [4,6,7]. Vectors are columns unless explicitly written as rows. For a vector v, $v[j]$ denotes its jth entry. For a matrix A, $A[j,:]$ denotes its jth row. The symbols T and $\dagger$ denote transpose and pseudoinverse, respectively. The identity matrix of size m is I_m.

Definition 1 (ReLU FCN). *An FCN $f_\theta : \mathbb{R}^{d_0} \to \mathbb{R}^{d_{k+1}}$ has k hidden layers of widths $d_1, \ldots, d_k$. It composes affine maps with componentwise ReLU, $\sigma(r) = \max\{r, 0\}$ [26]:*

$$f_\theta = f_{k+1} \circ \sigma \circ \cdots \circ \sigma \circ f_1,$$
$$f_j(x) = W^{(j)}x + b^{(j)}, \qquad 1 \le j \le k+1.$$

Here, $W^{(j)} \in \mathbb{R}^{d_j \times d_{j-1}}$ and $b^{(j)} \in \mathbb{R}^{d_j}$. Each hidden neuron applies ReLU to one scalar preactivation. The depth is k, and the total hidden-neuron count is $N_{\text{hid}} = \sum_{j=1}^{k} d_j$.

We write FCN architectures as d_0-$\cdots$-d_{k+1}. Exponents count consecutive layers of equal width: 20-10^3-1 = 20-10-10-10-1. An RNN has n recurrent layers of widths $d_1, \ldots, d_n$. A single-layer RNN has one recurrent layer, excluding the output layer. Its input is $X = (x^1, \ldots, x^\tau) \in \mathcal{X}_\tau := (\mathbb{R}^{d_0})^\tau$. We denote its architecture by d_0-d_1-$\cdots$-d_n-1, including one scalar output. Exponents count recurrent layers; τ counts time steps. Table 1 reports n and τ separately.

Definition 2 (Observation Settings). *For our scalar binary classifiers, the query oracles are [5]*

$$\mathcal{O}_{\text{raw}}(X) = f(X), \qquad \mathcal{O}_{\text{hard}}(X) = \mathbf{1}[f(X) > 0].$$

The raw-output setting exposes the logit; the hard-label setting exposes only its binary decision.

Definition 3 (Extended Functional Equivalence: Absolute [8] and Relative [23]). *Let f and $\widetilde{f}$ denote the scalar logits of the target and extracted RNNs. Let $\mathcal{S}$ be a finite nonempty set of input sequences. Each sequence $X \in \mathcal{S}$ has the form $(x^1, \ldots, x^\tau)$, where $x^t \in \mathbb{R}^{d_0}$. Fix $\varepsilon \ge 0$, $\delta \in [0, 1]$, and one scale $c > 0$ for all inputs. In the raw-output setting, $c = 1$. All probabilities below use uniform sampling from the indicated set. Absolute (ε, δ) functional equivalence requires*

$$\Pr_{X \in \mathcal{S}}\left[|\widetilde{f}(X) - cf(X)| \le \varepsilon\right] \ge 1 - \delta.$$

For nonempty $\mathcal{S}_{\neq 0} := \{X \in \mathcal{S} : f(X) \neq 0\}$, following [23], relative (ε, δ) functional equivalence requires

$$\Pr_{X \in \mathcal{S}_{\neq 0}}\left[\frac{|\widetilde{f}(X) - cf(X)|}{|cf(X)|} \le \varepsilon\right] \ge 1 - \delta.$$

Definition 4 (Critical Point [7]). *Let $z(x)$ denote a neuron's input features, with preactivation $w^\mathsf{T} z(x) + b$. An input x is critical for this neuron when its preactivation is zero. For $w \neq 0$, the critical hyperplane in feature space is $\{z : w^\mathsf{T} z + b = 0\}$. It separates the neuron's active and inactive regions.*

Definition 5 (Dual Point [6]). *A dual point is a critical point that also lies on the output decision boundary. The decision boundary separates input regions with different predicted labels.*

2.2 Functionally Equivalent Extraction

Existing FCN extraction methods use signature recovery and sign recovery [4,5,6,7]. They locate critical points in the raw-output setting or dual points in the hard-label setting. These points are clustered by neuron. For preactivation $w^\mathsf{T} z + b$, a signature is $\bar{w} = \alpha w$, where $\alpha \neq 0$ [6]. Critical points determine the bias at the same scale: $\bar{b} = \alpha b$. Sign recovery determines whether the recovered weights and bias need to be jointly negated. Positive scaling remains unconstrained.

Recovered neurons may differ in ordering and positive scaling [23]. Write $S_i = P_i Q_i$, where P_i is a permutation matrix and Q_i is positive diagonal. Set $S_0 = I_{d_0}$. A hat denotes quantities in these recovered coordinates, with $\widehat{h}^{(i)} = S_i h^{(i)}$. Here, $h^{(i)}$ denotes the hidden activation vector. Under exact recovery, the corresponding hidden-layer parameters satisfy

$$\widehat{W}^{(i)} = S_i W^{(i)} S_{i-1}^{-1}, \qquad \widehat{b}^{(i)} = S_i b^{(i)}, \qquad 1 \leq i \leq k.$$

For a scalar output, the output-layer parameters satisfy

$$\widehat{W}^{(k+1)} = c W^{(k+1)} S_k^{-1}, \qquad \widehat{b}^{(k+1)} = c b^{(k+1)}.$$

Together, these relations give $\widehat{f} = cf$. Each S_i has exactly one positive entry per row and column. All other entries vanish; such matrices are called positive monomial. They commute with ReLU: $\sigma(S_i r) = S_i \sigma(r)$ [31, Proposition 3.4].

2.3 Recurrent Neural Networks

We consider unidirectional many-to-one ReLU RNNs, as illustrated in Fig. 2. At layer ℓ, the state is $h_t^{(\ell)} \in \mathbb{R}^{d_\ell}$. Every sequence starts from $h_0^{(\ell)} = \mathbf{0}$. The first recurrent layer updates as follows:

$$h_t^{(1)} = \sigma\left(W_x^{(1)} x^t + W_h^{(1)} h_{t-1}^{(1)} + b^{(1)}\right), \qquad 1 \leq t \leq \tau. \tag{1}$$

For every higher recurrent layer $2 \leq \ell \leq n$, the state update is

$$h_t^{(\ell)} = \sigma\left(W_x^{(\ell)} h_t^{(\ell-1)} + W_h^{(\ell)} h_{t-1}^{(\ell)} + b^{(\ell)}\right), \qquad 1 \leq t \leq \tau.$$

Here, $W_x^{(1)} \in \mathbb{R}^{d_1 \times d_0}$, $W_x^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ for $\ell \geq 2$, $W_h^{(\ell)} \in \mathbb{R}^{d_\ell \times d_\ell}$, and $b^{(\ell)} \in \mathbb{R}^{d_\ell}$. We call $W_x^{(\ell)}$ the input weight matrix of layer ℓ. It maps x^t when $\ell = 1$ and $h_t^{(\ell-1)}$ when $\ell \geq 2$. We call $W_h^{(\ell)}$ the recurrent weight matrix. It propagates the hidden state of layer ℓ across time.

The classifier outputs the scalar logit $f(X) = W_o h_\tau^{(n)} + b_o$. Here, $W_o \in \mathbb{R}^{1 \times d_n}$ and $b_o \in \mathbb{R}$. Its predicted label is $\mathbf{1}[f(X) > 0]$.

Definition 6 (Recurrent Consistency). *Let $S_\ell \in \mathbb{R}^{d_\ell \times d_\ell}$ be positive monomial, with $S_0 = I_{d_0}$. Recurrent consistency requires these matrices to be independent of time and input. The recovered parameters must satisfy*

$$\widehat{W}_x^{(\ell)} = S_\ell W_x^{(\ell)} S_{\ell-1}^{-1}, \quad \widetilde{W}_h^{(\ell)} = S_\ell W_h^{(\ell)} S_\ell^{-1}, \quad \widetilde{b}^{(\ell)} = S_\ell b^{(\ell)}, \quad 1 \leq \ell \leq n.$$

Input weights define the reference coordinates and retain hats. Tildes denote recurrent parameters and states aligned to those coordinates.

With zero initial states, these relations imply

$$\widetilde{h}_t^{(\ell)} = S_\ell h_t^{(\ell)}, \qquad 1 \leq \ell \leq n, \quad 0 \leq t \leq \tau,$$

for every input sequence. The same parameters can therefore be reused across time. With output parameters

$$\widehat{W}_o = c W_o S_n^{-1}, \qquad \widehat{b}_o = c b_o,$$

the aligned RNN satisfies $\widetilde{f}(X) = c f(X)$ for every $X \in \mathcal{X}_\tau$.

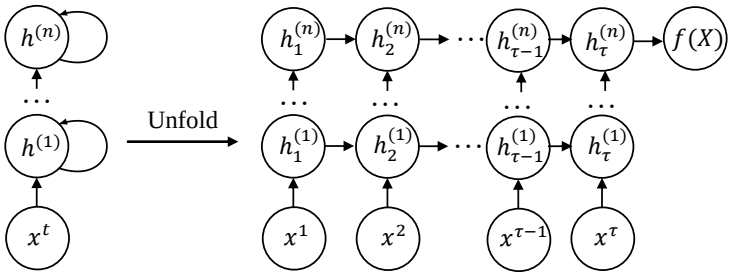


Fig. 2: A stacked RNN with n recurrent layers and an unfolded many to one computation. In the folded representation, $h^{(\ell)}$ denotes the recurrent state of layer ℓ without a specific time index. In the unfolded representation, x^t and $h_t^{(\ell)}$ denote the input and hidden state at time t, respectively.

2.4 Adversary Capabilities and Assumptions

The attacker adaptively queries either oracle in Definition 2. Hidden states remain unobservable.

We make the following assumptions. They follow cryptanalytic extraction models [4,7].

- **ReLU activation.** The target uses ReLU in every recurrent layer.
- **Arithmetic.** Exact recovery arguments use exact arithmetic. Experiments use 64-bit floating-point arithmetic.
- **Known architecture.** The attacker knows n, every layer width, and the sequence length τ.
- **Unrestricted inputs.** The attacker controls every real-valued vector at every time step.

3 Attacks on Single-Layer RNNs

This section introduces attacks on single-layer RNNs in the raw-output and hard-label settings. To address the challenges that computation across time steps poses for RNN recovery, we construct two equivalent FCNs whose depths are independent of the sequence length. This divides parameter recovery into two stages: recovering input weights and output parameters, followed by recurrent weights and biases. To resolve the mismatches in neuron ordering and scaling that prevent parameter reuse, we propose RPA. It constructs equations involving folded biases and uses them to align the recovered parameters for reuse across time.

3.1 Equivalent FCNs for Single-Layer RNNs

To extract an RNN with one recurrent layer, a straightforward reduction varies x^1 and sets $x^2, \ldots, x^\tau$ to the zero vector, yielding an equivalent FCN of depth τ. This reduction has two limitations. First, the FCN depth grows with the sequence length, making extraction increasingly difficult. Existing end-to-end cryptanalytic extraction attacks have been demonstrated on networks with at most eight hidden layers [23]. Second, the recovered parameters can reproduce the network's outputs on sequences of the form $(x^1, \mathbf{0}, \ldots, \mathbf{0})$, but do not guarantee equivalence on arbitrary input sequences because the input and recurrent weights recovered through FCN extraction may use inconsistent neuron ordering and scaling. We instead vary the final or penultimate input, fixing all others to zero. The resulting models $M_1^{(1)}$ and $M_2^{(1)}$ have one and two hidden layers, respectively. Both depths are independent of the sequence length. $M_1^{(1)}$ recovers input weights $W_x^{(1)}$, a folded bias, and output parameters (W_o, b_o). $M_2^{(1)}$ reuses the recovered input weights to recover $(W_h^{(1)}, b^{(1)})$. RPA then aligns the recurrent parameters for reuse across time.

Fig. 3(a) shows the construction of $M_1^{(1)}$. Its input sequence is $(\mathbf{0}, \ldots, \mathbf{0}, x^\tau)$: only the final input x^τ varies, while all preceding inputs are zero vectors, represented by the green input nodes. The blue box represents the computation induced by the preceding zero input vectors. Its folded bias vector is denoted by $\beta_\tau^{(1)}$. This vector is independent of x^τ. We define this folded bias using the hidden states generated by the zero input prefix.

Let $\bar{h}_t^{(1)}$ denote the hidden state after t zero input vectors, starting from $\bar{h}_0^{(1)} = \mathbf{0}$. It is computed using Equation (1) with $x^s = \mathbf{0}$ for $1 \le s \le t$. At each of the final two steps, the folded bias combines the recurrent contribution of the preceding hidden state generated by zero input vectors with the original bias:

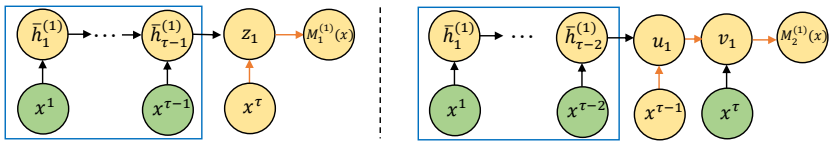
$$\beta_t^{(1)} := W_h^{(1)} \bar{h}_{t-1}^{(1)} + b^{(1)}, \qquad t \in \{\tau - 1, \tau\}.$$

The attacker recovers $\widehat{\beta}_t^{(1)}$ through extraction of the corresponding equivalent FCN.

Following the computational flow of $M_1^{(1)}$ along the orange arrows in Fig. 3, we obtain an equivalent FCN with one hidden layer:

$$z_1 := \sigma\left(W_x^{(1)} x + \beta_\tau^{(1)}\right), \quad M_1^{(1)}(x) := f(\mathbf{0}, \ldots, \mathbf{0}, \mathbf{0}, x) = W_o z_1 + b_o. \tag{2}$$

Fig. 4(a) shows this structure, with input dimension d_0, hidden width d_1, and one output. Recovery yields equivalent copies of $W_x^{(1)}$, $\beta_\tau^{(1)}$, and (W_o, b_o).

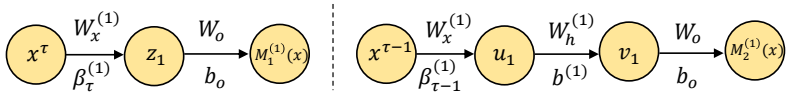


(a) (b)

Fig. 3: Controlled input sequences for a single-layer RNN. (a) varies x^τ. (b) varies $x^{\tau-1}$ and fixes $x^\tau = \mathbf{0}$. Green input nodes are fixed to zero; yellow input nodes are variable. The blue boxes enclose the computation induced by prefix inputs held constant across queries. These states are independent of the variable input. We choose zero prefix inputs for simplicity and assume $h_0^{(\ell)} = \mathbf{0}$ for every sequence. Orange arrows indicate paths affected by the variable input.

We construct $M_2^{(1)}$ using the controlled input sequence $(\mathbf{0}, \ldots, \mathbf{0}, x, \mathbf{0})$, as illustrated in Fig. 3(b). Only the penultimate input $x^{\tau-1} = x$ varies; all other inputs are zero vectors. The blue box represents the sequence of zero input vectors through time $\tau - 2$, which yields the folded bias vector $\beta_{\tau-1}^{(1)}$. Following the indicated data flow, the resulting model is

$$\begin{aligned} u_1 &:= \sigma\left(W_x^{(1)} x + \beta_{\tau-1}^{(1)}\right), & v_1 &:= \sigma\left(W_h^{(1)} u_1 + b^{(1)}\right), \\ M_2^{(1)}(x) &:= f(\mathbf{0}, \ldots, \mathbf{0}, x, \mathbf{0}) = W_o v_1 + b_o. \end{aligned} \tag{3}$$



(a) (b)

Fig. 4: Equivalent FCNs induced by the two controlled inputs: (a) $M_1^{(1)}$; (b) $M_2^{(1)}$.

Hence, $M_2^{(1)}$ is an FCN with two hidden layers. Fig. 4(b) shows this structure. Its input dimension is d_0. Both hidden layers have width d_1. The model has one output.

We recover $M_1^{(1)}$ first, then fix its recovered input weights $\widehat{W}_x^{(1)}$ in $M_2^{(1)}$. The raw-output setting uses FCN signature and sign recovery [7,4]. The hard-label setting uses methods based on decision boundaries and dual points [6,5]. Both settings supply signed parameters to the same RPA procedure.

The following identities assume exact FCN recovery, including signs. We next compare the recovered parameter representations of the two models to identify the alignment of neuron ordering and scaling required for reuse. Recovered logits agree with $M_1^{(1)}$ up to an output scale. The recovered hidden neurons may be permuted relative to those in the target model. Each neuron's incoming weights and bias may also be multiplied by its own positive factor. Let $T := P_1 Q_1$, where P_1 is a permutation matrix and Q_1 is a positive diagonal matrix. Let $c = 1$ in the raw-output setting. In the hard-label setting, $c > 0$ denotes the unavoidable output scale. Recovery of $M_1^{(1)}$ gives

$$\widehat{W}_x^{(1)} = T W_x^{(1)}, \quad \widehat{\beta}_\tau^{(1)} = T\beta_\tau^{(1)}, \quad \widehat{W}_o = c W_o T^{-1}, \quad \widehat{b}_o = c b_o.$$

First-layer critical points matched to $\widehat{W}_x^{(1)}$ determine $\widehat{\beta}_{\tau-1}^{(1)} = T\beta_{\tau-1}^{(1)}$. Hard-label recovery uses corresponding dual points. Equation (5) gives the analogous calculation for probe biases. With this layer fixed, recovery of the second hidden layer returns

$$\widehat{W}_h^{(1)} = R W_h^{(1)} T^{-1}, \qquad \widehat{b}^{(1)} = R b^{(1)}. \tag{4}$$

Here, $R = P_2 Q_2$ is an independently recovered transformation of neuron ordering and scaling. The index 2 refers to the second hidden layer of $M_2^{(1)}$. Both T and R belong to $\mathbb{R}^{d_1 \times d_1}$. The input and output neuron ordering and scaling of $\widehat{W}_h^{(1)}$ are specified by T and R, respectively. Suppose that $\widehat{h}_{t-1}^{(1)} = T h_{t-1}^{(1)}$. Direct reuse would give

$$\widehat{W}_x^{(1)} x^t + \widehat{W}_h^{(1)} \widehat{h}_{t-1}^{(1)} + \widehat{b}^{(1)} = T W_x^{(1)} x^t + R\left(W_h^{(1)} h_{t-1}^{(1)} + b^{(1)}\right), \qquad 1 \le t \le \tau.$$

The two terms use different neuron ordering and scaling when $R \neq T$. Hence, $(\widehat{W}_h^{(1)}, \widehat{b}^{(1)})$ generally cannot be reused directly across time.

We therefore propose the Recurrent Parameter Alignment (RPA) method to resolve this mismatch by computing $L := TR^{-1}$. Section 3.2 describes this procedure. It then aligns the recovered recurrent pair as

$$\widetilde{W}_h^{(1)} = L\widehat{W}_h^{(1)} = TW_h^{(1)}T^{-1}, \qquad \widetilde{b}^{(1)} = L\widehat{b}^{(1)} = Tb^{(1)}.$$

These relations satisfy Definition 6 when $n = 1$, with T specifying the hidden coordinates. The following theorem establishes equivalence under exact recovery and alignment.

Theorem 1 (Functional Equivalence of Recovered Single-Layer RNNs). *Let $T = P_1Q_1$ be a positive monomial matrix, as defined above. Suppose the aligned parameters satisfy*

$$\widehat{W}_x^{(1)} = TW_x^{(1)}, \ \widetilde{W}_h^{(1)} = TW_h^{(1)}T^{-1}, \ \widetilde{b}^{(1)} = Tb^{(1)}, \ \widehat{W}_o = cW_oT^{-1}, \ \widehat{b}_o = cb_o.$$

Here, $c = 1$ in the raw-output setting and $c > 0$ in the hard-label setting. Let $\widetilde{f}$ denote the aligned RNN. Both models use zero initial states. Therefore, $\widetilde{h}_0^{(1)} = Th_0^{(1)} = \mathbf{0}$. Then every input sequence $X = (x^1, \ldots, x^\tau)$ satisfies

$$\widetilde{h}_t^{(1)} = Th_t^{(1)} \quad \text{for } 0 \le t \le \tau, \qquad \widetilde{f}(X) = cf(X).$$

Consequently, both models have identical scores in the raw-output setting and identical labels in the hard-label setting.

Proof. Zero initialization gives $\widetilde{h}_0^{(1)} = Th_0^{(1)} = \mathbf{0}$. Assume $\widetilde{h}_{t-1}^{(1)} = Th_{t-1}^{(1)}$. Since positive monomial matrices commute with ReLU (Section 2.2),

$$\widetilde{h}_t^{(1)} = \sigma\Big(T\left[W_x^{(1)}x^t + W_h^{(1)}h_{t-1}^{(1)} + b^{(1)}\right]\Big) = Th_t^{(1)}.$$

Induction establishes this relation at every time step. The output therefore satisfies

$$\widetilde{f}(X) = cW_oT^{-1}\widetilde{h}_\tau^{(1)} + cb_o = cf(X).$$

Positive c preserves binary labels.

3.2 Recurrent Parameter Alignment for Single-Layer RNNs

This section explains how RPA achieves recurrent consistency. Its core idea is to exploit mathematical relations involving the folded biases of the two equivalent models.

We first analyze the mathematical relation between the recovered folded biases in the first layers of the two equivalent models. Recovery gives

$$\widehat{\beta}_{\tau-1}^{(1)} = T\beta_{\tau-1}^{(1)}, \qquad \widehat{\beta}_\tau^{(1)} = T\beta_\tau^{(1)}.$$

The folded biases satisfy $\beta_\tau^{(1)} = W_h^{(1)}\sigma(\beta_{\tau-1}^{(1)}) + b^{(1)}$. Therefore, applying $\widehat{W}_h^{(1)}$ and $\widehat{b}^{(1)}$ to $\sigma(\widehat{\beta}_{\tau-1}^{(1)})$ yields another representation of $\beta_\tau^{(1)}$, with different neuron ordering and scaling from $\widehat{\beta}_\tau^{(1)}$:

$$s_0 := \widehat{W}_h^{(1)}\sigma\Big(\widehat{\beta}_{\tau-1}^{(1)}\Big) + \widehat{b}^{(1)} = RW_h^{(1)}T^{-1}\sigma\Big(T\beta_{\tau-1}^{(1)}\Big) + Rb^{(1)} = R\beta_\tau^{(1)}.$$

The last equality uses $\sigma(Tr) = T\sigma(r)$. Thus, $\widehat{\beta}_\tau^{(1)} = TR^{-1}s_0$. This single pair generally cannot identify the monomial map. Several coordinate permutations may satisfy this relation. Positive scales can fit paired nonzero entries with the same sign.

Therefore, RPA constructs additional equations using controlled probes to recover the alignment matrix $L = TR^{-1}$.

Constructing additional folded biases. The preceding analysis compares $T\beta_\tau^{(1)}$ and $R\beta_\tau^{(1)}$, two representations of the folded bias of $M_1^{(1)}$. RPA extends this idea by constructing models similar to $M_1^{(1)}$, with the penultimate input fixed to a random vector ξ_k instead of the zero vector. Each choice produces a folded bias r_k that depends on ξ_k. Using $\widehat{W}_x^{(1)}$, we can recover Tr_k from the probe model with the same neuron ordering and scaling. If we can compute Rr_k using the recovered parameters, these representations provide an additional equation for determining L.

We start with K_0 independent Gaussian probes. Additional probes may be collected when alignment remains unresolved. Let $K \geq K_0$ denote the final number of probes used for alignment. For each $1 \leq k \leq K$, independently sample $\xi_k \sim \mathcal{N}(\mathbf{0}, I_{d_0})$. Fix $x^{\tau-1} = \xi_k$ and vary $x^\tau = x$. Keep every earlier input equal to the zero vector. Each probe ξ_k defines a family of queries $(\mathbf{0}, \ldots, \mathbf{0}, \xi_k, x)$, in which x varies to locate the required critical or dual points. Let $\alpha_k^{(1)}$ denote the first layer preactivation at time $\tau-1$. Fig. 5 illustrates this data flow. Each probe defines the following shallow FCN:

$$\alpha_k^{(1)} := W_x^{(1)}\xi_k + \beta_{\tau-1}^{(1)}, \qquad r_k := W_h^{(1)}\sigma\Big(\alpha_k^{(1)}\Big) + b^{(1)},$$
$$M_{\xi_k}^{(1)}(x) := W_o\sigma\Big(W_x^{(1)}x + r_k\Big) + b_o.$$

The probes provide two representations of each folded bias r_k. First, associate a queried critical point $x_{j,k}^*$ with the known row $\widehat{W}_x^{(1)}[j,:]$. In the hard-label setting, use a corresponding dual point. The point's zero preactivation fixes the bias in these coordinates:

$$\widehat{r}_k[j] = -\widehat{W}_x^{(1)}[j,:]x_{j,k}^*, \qquad \widehat{r}_k = Tr_k. \tag{5}$$

Second, the recovered recurrent parameters provide another representation. Using $\widehat{W}_x^{(1)}$ and $\widehat{\beta}_{\tau-1}^{(1)}$, we can compute

$$g_k := \sigma\Big(\widehat{W}_x^{(1)}\xi_k + \widehat{\beta}_{\tau-1}^{(1)}\Big) = T\sigma\Big(\alpha_k^{(1)}\Big).$$

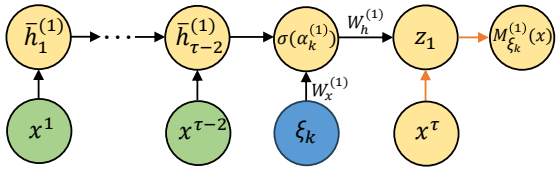


Fig. 5: Controlled probe for folded bias recovery in a single-layer RNN. The penultimate input is fixed to ξ_k. The final input x^τ remains variable. Green input nodes are fixed to zero, the blue input node is the fixed probe, and the yellow input node is variable. The fixed probe induces a folded bias independent of the variable input.

This equality uses $\sigma(Tr) = T\sigma(r)$. The recovered recurrent pair in Equation (4) predicts

$$s_k := \widehat{W}_h^{(1)} g_k + \widehat{b}^{(1)} = R r_k. \tag{6}$$

Combining Equations (5) and (6) yields $\widehat{r}_k = L s_k$, where $L = TR^{-1}$.

Neuron ordering and scaling recovery. The next step is to express the folded bias equations in matrix form. We construct two matrices by arranging s_i, $0 \le i \le K$, and $\widehat{\beta}_\tau^{(1)}, \widehat{r}_1, \ldots, \widehat{r}_K$ as their respective columns:

$$\Phi := \left[s_0 \; s_1 \cdots s_K \right], \qquad B := \left[\widehat{\beta}_\tau^{(1)} \; \widehat{r}_1 \cdots \widehat{r}_K \right] = TR^{-1}\Phi.$$

Here, $\Phi, B \in \mathbb{R}^{d_1 \times (K+1)}$.

Since T and R are positive monomial matrices, closure under inversion and multiplication implies that $L = TR^{-1}$ is also positive monomial (Lemma 2 and Theorem 3 of [25]). By definition, each row of L has exactly one nonzero entry, which is positive. If $L[j,p] = \lambda_j > 0$, then $\widehat{r}_k = L s_k$ gives $\widehat{r}_k[j] = \lambda_j s_k[p]$ for every probe k. Collecting these values across probes forms proportional rows, allowing neuron matching by comparing row proportionality.

Thus, a correct match satisfies $B[j,:] = \lambda_j \Phi[p,:]$. For unique identification, assume that all rows of Φ are nonzero and pairwise nonproportional. If Φ contains a zero row, collect additional probes before evaluating $\rho_{p,j}$. If a zero row persists, leave alignment unresolved. For neuron p before alignment and its candidate match j, define the scale and residual:

$$\rho_{p,j} := \frac{\Phi[p,:] B[j,:]^\mathsf{T}}{\Phi[p,:] \Phi[p,:]^\mathsf{T}}, \qquad C_{p,j} := \frac{\| B[j,:] - \rho_{p,j} \Phi[p,:] \|_2}{\max\{ \| B[j,:] \|_2, \eta \}}.$$

Here, $\rho_{p,j}$ is the least squares scale, and $C_{p,j}$ is the normalized residual after scaling. For a correct match, $\rho_{p,j} > 0$ equals the corresponding entry of TR^{-1}. The floor $\eta > 0$ only stabilizes residual normalization. Let $C = [C_{p,j}] \in \mathbb{R}^{d_1 \times d_1}$ be the assignment cost matrix. We set $C_{p,j} = +\infty$ whenever $\rho_{p,j} \le 0$, excluding these candidates. The Hungarian algorithm minimizes the total cost over admissible neuron bijections [19]. If no admissible bijection exists, alignment remains

unresolved. Let $p = \pi(j)$ denote the selected match for neuron j. The selected scales define $L \in \mathbb{R}^{d_1 \times d_1}$ by

$$L[j,p] := \begin{cases} \rho_{p,j}, & p = \pi(j), \\ 0, & \text{otherwise.} \end{cases}$$

Applying L to Equation (4) gives

$$\widetilde{W}_h^{(1)} = L\widehat{W}_h^{(1)} = TW_h^{(1)}T^{-1}, \qquad \widetilde{b}^{(1)} = L\widehat{b}^{(1)} = Tb^{(1)}. \tag{7}$$

Under recovery and correct alignment, Equation (7) satisfies Theorem 1.

We also consider the special case where two rows of Φ are proportional. By Equation (6), one sufficient cause is that the corresponding rows of $[\widehat{W}_h^{(1)}\ \widehat{b}^{(1)}]$ are positively proportional, making their values proportional for every ξ_k. In this case, uniquely determining the permutation within this group is unnecessary: appropriate rescaling yields identical aligned parameters and preserves recurrent consistency. Proportionality on finitely many probes alone does not establish this parameter relation. If the parameter rows lack this relation, collect additional probes.

Complexity. This analysis excludes the construction of Φ and B. Given these matrices, all pairwise costs require $O(Kd_1^2)$ operations. The Hungarian assignment requires $O(d_1^3)$ operations. Applying the recovered transformation requires $O(d_1^2)$ operations. Thus, the alignment stage costs $O(Kd_1^2 + d_1^3)$.

4 Attacks on Stacked RNNs

We reuse the controlled input placements from Section 3.1 to construct $M_1^{(n)}$ and $M_2^{(n)}$. We first recover the input weights from the equivalent FCN $M_1^{(n)}$. Using these weights, we recover the previous branch biases of $M_2^{(n)}$ in the same coordinates. Recurrent Parameter Alignment (RPA) then aligns the first recurrent layer and recovers its signs. For subsequent layers, Recurrent Parameter Recovery (RPR) recovers recurrent parameters using the known input weights as anchors. Folded Bias Difference (FBD) supplements RPR for difficult neurons.

4.1 Equivalent Feedforward Models for Stacked RNNs

We reuse the controlled input placements from Section 3.1, with $\tau \geq 2$. For $n \geq 2$, $M_1^{(n)}$ remains an FCN, whereas $M_2^{(n)}$ contains two coupled branches. Both models exactly represent the target on their respective input families.

Construction of $M_1^{(n)}$. Set $x^\tau = x$ and all earlier inputs to zero; see Fig. 7(a). The resulting zero-input states $\bar{h}_t^{(\ell)}$ satisfy $\bar{h}_0^{(\ell)} = \mathbf{0}$. For $1 \leq t < \tau$, their recurrence

is

$$\bar{h}_t^{(1)} = \sigma\Big(W_h^{(1)}\bar{h}_{t-1}^{(1)} + b^{(1)}\Big),$$
$$\bar{h}_t^{(\ell)} = \sigma\Big(W_x^{(\ell)}\bar{h}_t^{(\ell-1)} + W_h^{(\ell)}\bar{h}_{t-1}^{(\ell)} + b^{(\ell)}\Big), \quad 2 \le \ell \le n.$$

For $1 \le \ell \le n$ and $1 \le t \le \tau$, define the folded bias $\beta_t^{(\ell)} = W_h^{(\ell)}\bar{h}_{t-1}^{(\ell)} + b^{(\ell)}$.

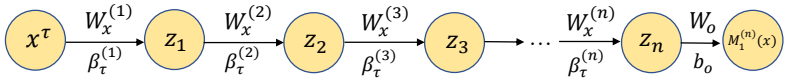


Fig. 6: Equivalent FCN $M_1^{(n)}$ induced by $x^\tau = x$ and $x^t = \mathbf{0}$ for $t < \tau$.

The fixed prefix leaves the following variable computation:

$$z_1 = \sigma\Big(W_x^{(1)}x + \beta_\tau^{(1)}\Big), \qquad z_\ell = \sigma\Big(W_x^{(\ell)}z_{\ell-1} + \beta_\tau^{(\ell)}\Big), \quad 2 \le \ell \le n,$$
$$M_1^{(n)}(x) = W_o z_n + b_o.$$

Thus, $M_1^{(n)}(x) = f(\mathbf{0}, \ldots, \mathbf{0}, x)$ is an FCN with n hidden layers. Its recovery yields the input weights and folded biases. We use $M_2^{(n)}$ to recover the recurrent parameters.

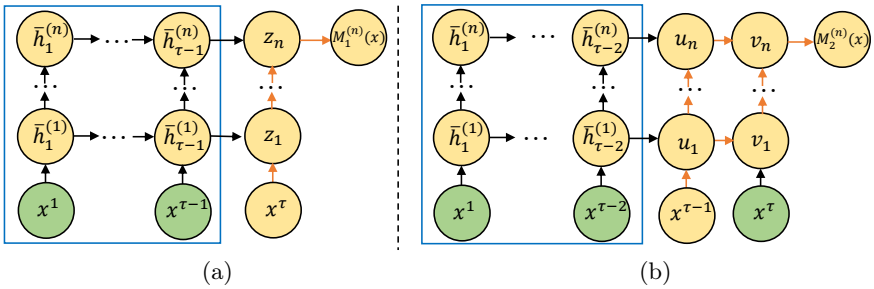


Fig. 7: Controlled inputs for a stacked RNN: (a) varies x^τ; (b) varies $x^{\tau-1}$ with $x^\tau = \mathbf{0}$. Green input nodes are fixed to zero; yellow input nodes are variable. The blue boxes enclose the computation induced by prefix inputs held constant across queries. These states are independent of the variable input. We choose zero prefix inputs for simplicity and assume $h_0^{(\ell)} = \mathbf{0}$ for every sequence. Orange arrows indicate paths affected by the variable input.

Construction of $M_2^{(n)}$. Set $x^{\tau-1} = x$ and all other inputs to zero; see Fig. 7(b). Let u_ℓ denote the layer-ℓ state at time $\tau - 1$. Then

$$u_1 = \sigma\Big(W_x^{(1)} x + \beta_{\tau-1}^{(1)}\Big), \qquad u_\ell = \sigma\Big(W_x^{(\ell)} u_{\ell-1} + \beta_{\tau-1}^{(\ell)}\Big), \quad 2 \le \ell \le n.$$

Let v_ℓ denote the state of layer ℓ at time τ. Since $x^\tau = \mathbf{0}$,

$$v_1 = \sigma\Big(W_h^{(1)} u_1 + b^{(1)}\Big), \quad v_\ell = \sigma\Big(W_x^{(\ell)} v_{\ell-1} + W_h^{(\ell)} u_\ell + b^{(\ell)}\Big), \; 2 \le \ell \le n,$$

$M_2^{(n)}(x) = W_o v_n + b_o$.

The states $(u_1, \ldots, u_n)$ form the previous branch; $(v_1, \ldots, v_n)$ form the current branch. Fig. 8 shows their coupling through the recurrent weights. The model satisfies $M_2^{(n)}(x) = f(\mathbf{0}, \ldots, \mathbf{0}, x, \mathbf{0})$.

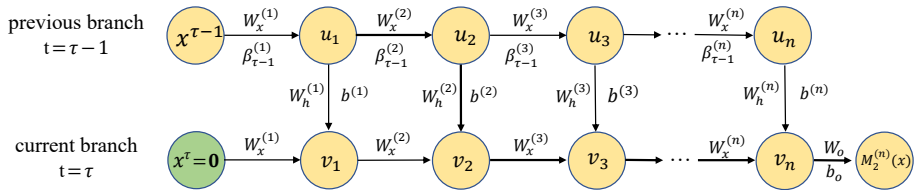


Fig. 8: Equivalent feedforward model $M_2^{(n)}$. The upper and lower branches contain the states at times $\tau - 1$ and τ, respectively. The final input is fixed at $x^\tau = \mathbf{0}$.

We can recover $M_1^{(n)}$ first using existing FCN extraction methods. For each layer $1 \le \ell \le n$, let the positive monomial matrix $S_\ell = P_\ell Q_\ell$ represent the recovered neuron ordering and scaling. Under exact recovery, the parameters of $M_1^{(n)}$ satisfy

$$\begin{aligned} \widehat{W}_x^{(1)} &= S_1 W_x^{(1)}, & \widehat{\beta}_\tau^{(1)} &= S_1 \beta_\tau^{(1)}, \\ \widehat{W}_x^{(\ell)} &= S_\ell W_x^{(\ell)} S_{\ell-1}^{-1}, & \widehat{\beta}_\tau^{(\ell)} &= S_\ell \beta_\tau^{(\ell)}, \qquad 2 \le \ell \le n, \\ \widehat{W}_o &= c W_o S_n^{-1}, & \widehat{b}_o &= c b_o. \end{aligned} \tag{8}$$

Here, $c > 0$, with $c = 1$ in the raw-output setting.

During the recovery of $M_2^{(n)}$, the previous branch reuses the recovered matrices $\widehat{W}_x^{(\ell)}$ from $M_1^{(n)}$. Within the previous branch, the neuron ordering and scaling transformations of adjacent layers cancel. The recovered matrices can therefore be reused directly. We recover these biases layer by layer using the corresponding zero-preactivation equations. Raw-output queries supply critical points; hard-label queries supply dual points. At a point x^* associated with neuron j,

$$\widehat{\beta}_{\tau-1}^{(\ell)}[j] = -\widehat{W}_x^{(\ell)}[j, :]\, \widetilde{u}_{\ell-1}(x^*), \qquad \widetilde{u}_0(x^*) = x^*.$$

Here, $\widetilde{u}_{\ell-1}$ is computed from the recovered previous branch prefix. They satisfy $\widehat{\beta}^{(\ell)}_{\tau-1} = S_\ell \beta^{(\ell)}_{\tau-1}$ for $1 \le \ell \le n$. The resulting previous branch states satisfy $\widetilde{u}_\ell = S_\ell u_\ell \in \mathbb{R}^{d_\ell}$.

We now consider the recovery of the current branch in $M_2^{(n)}$. The model remains piecewise affine. Since the first current-branch layer receives only u_1, rather than the joint features $(v_{\ell-1}, u_\ell)$ of subsequent layers, existing methods can be used to recover its signatures. However, the coupled downstream branches still complicate sign recovery. For the first layer of the current branch, the parameters after signature recovery can be written as

$$\widehat{W}_h^{(1)} = D_s T_1 W_h^{(1)} S_1^{-1}, \qquad \widehat{b}^{(1)} = D_s T_1 b^{(1)},$$

Here, $T_1, D_s \in \mathbb{R}^{d_1 \times d_1}$. The matrix T_1 is positive monomial. The matrix D_s is diagonal with entries in $\{-1, 1\}$. We extend RPA to jointly recover the unknown signs and align neuron ordering and scaling through folded bias relations, thereby establishing recurrent consistency. The aligned parameters take the form

$$\widetilde{W}_h^{(1)} = S_1 W_h^{(1)} S_1^{-1}, \qquad \widetilde{b}^{(1)} = S_1 b^{(1)}. \tag{9}$$

After RPA, recover layers $2, \dots, n$ sequentially. At layer ℓ, reuse $\widehat{W}_x^{(\ell)}$ once the preceding current-branch layers are in the reference coordinates. We propose RPR, which uses $\widehat{W}_x^{(\ell)}$ to guide clustering and recurrent parameter recovery while fixing neuron signs, ordering, and scaling. The resulting relations are

$$\widetilde{W}_h^{(\ell)} = S_\ell W_h^{(\ell)} S_\ell^{-1}, \quad \widetilde{b}^{(\ell)} = S_\ell b^{(\ell)}, \qquad 2 \le \ell \le n. \tag{10}$$

The following theorem establishes functional equivalence under the stated exact parameter relations.

Theorem 2 (Functional Equivalence of the Recovered Stacked RNN). *Suppose the parameters recovered from $M_1^{(n)}$ satisfy Equation (8). Those recovered from $M_2^{(n)}$ satisfy Equations (9) and (10). Let $\widetilde{f}$ use $\widehat{W}_x^{(\ell)}$, $\widetilde{W}_h^{(\ell)}$, and $\widetilde{b}^{(\ell)}$, with output parameters $(\widehat{W}_o, \widehat{b}_o)$. Both RNNs have zero initial states. Then, for every input sequence $X = (x^1, \dots, x^\tau)$,*

$$\widetilde{h}_t^{(\ell)} = S_\ell h_t^{(\ell)}, \qquad 1 \le t \le \tau, \quad 1 \le \ell \le n, \qquad \widetilde{f}(X) = c f(X).$$

Consequently, since $c > 0$,

$$\mathbf{1}\left[\widetilde{f}(X) > 0\right] = \mathbf{1}[f(X) > 0]\,. \tag{11}$$

In the raw-output setting, $c = 1$, so both RNNs produce identical scores.

The proof follows Theorem 1, with an additional induction over layers; see Appendix A.1.

4.2 Application of RPA to Stacked RNNs

We extend RPA to align the first recurrent layer and recover its signs. The coupled branches explain why conventional sign recovery does not apply directly.

Sign Recovery for the Current Branch of $M_2^{(n)}$. We briefly recall the FCN sign recovery methods SOE and Neuron Wiggle used in the raw-output setting [4]. SOE fits local output changes to target preactivation changes. With sufficient local rank and nonzero downstream coefficients for active neurons, the fitted zero coefficients identify inactive neurons. Neuron Wiggle maximizes the target neuron's change within the available space of control and compares output-change magnitudes on opposite sides of a critical point. In the hard-label setting, sign recovery methods [5,6] follow a similar idea by examining how changes in target-layer neurons affect the output decision boundary.

For $M_2^{(n)}$ in Fig. 8, varying $x^{\tau-1}$ affects the output not only through v_1, but also through paths along the previous branch that bypass v_1. These paths contribute additional output changes even when the target neuron in v_1 is inactive, interfering with both the zero-coefficient test and the magnitude comparison. These sign recovery methods therefore cannot be applied directly to the current branch.

Applying RPA to Stacked RNNs. As in Section 3.2, start with K_0 independent Gaussian probes. Collect additional probes when alignment remains unresolved. Let $K \geq K_0$ denote the final probe count, with $\xi_k \sim \mathcal{N}(\mathbf{0}, I_{d_0})$. We construct the model $M_{\xi_k}^{(n)}(x)$, as illustrated in Fig. 9. In trial k, fix $x^{\tau-1} = \xi_k$ and vary $x^\tau = x$. Keep every earlier input equal to the zero vector. Define

$$\alpha_k^{(1)} = W_x^{(1)}\xi_k + \beta_{\tau-1}^{(1)}, \qquad r_k^{(1)} = W_h^{(1)}\sigma(\alpha_k^{(1)}) + b^{(1)}. \tag{12}$$

The first current branch state is $z_1 = \sigma\left(W_x^{(1)}x + r_k^{(1)}\right)$. The folded bias $r_k^{(1)}$ is independent of x.

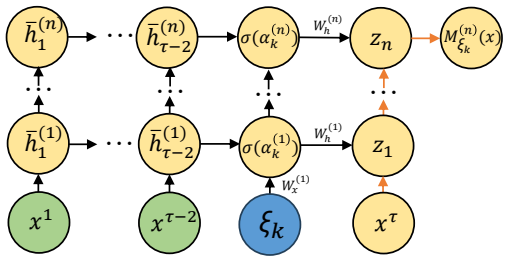


Fig. 9: Controlled probe for $M_{\xi_k}^{(n)}(x)$. The penultimate input is fixed to ξ_k. The final input x^τ remains variable. Green input nodes are fixed to zero, the blue input node is the fixed probe, and the yellow input node is variable. The fixed probe induces a folded bias independent of the variable input.

For $M_{\xi_k}^{(n)}$, we match the first layer weights to $\widehat{W}_x^{(1)}$ and recover the folded bias satisfying $\widehat{r}_k^{(1)} = S_1 r_k^{(1)}$. Meanwhile, the recovered folded bias $\widehat{\beta}_{\tau-1}^{(1)}$ and the unaligned recurrent parameters $\widehat{W}_h^{(1)} = D_s T_1 W_h^{(1)} S_1^{-1}$ and $\widehat{b}^{(1)} = D_s T_1 b^{(1)}$ give the prediction

$$\check{r}_k^{(1)} = \widehat{W}_h^{(1)} \sigma(\widehat{W}_x^{(1)} \xi_k + \widehat{\beta}_{\tau-1}^{(1)}) + \widehat{b}^{(1)} = D_s T_1 r_k^{(1)},$$

Here, D_s is the diagonal sign matrix to be recovered. We retain the probe $\xi_0 = \mathbf{0}$ as a separate baseline, as in Section 3.2. Its responses satisfy

$$\widehat{r}_0^{(1)} = \widehat{\beta}_{\tau}^{(1)}, \qquad \check{r}_0^{(1)} = \widehat{W}_h^{(1)} \sigma(\widehat{\beta}_{\tau-1}^{(1)}) + \widehat{b}^{(1)}.$$

Stack the baseline and probe responses as columns:

$$\Phi = [\check{r}_0^{(1)} \ \check{r}_1^{(1)} \ \cdots \ \check{r}_K^{(1)}], \qquad B = [\widehat{r}_0^{(1)} \ \widehat{r}_1^{(1)} \ \cdots \ \widehat{r}_K^{(1)}].$$

Both matrices belong to $\mathbb{R}^{d_1 \times (K+1)}$. The response relation is $B = L\Phi$. Here, $L = S_1 {T_1}^{-1} {D_s}^{-1}$. The matrix L is monomial: each row and column contains exactly one nonzero entry, which may have either sign. Matched rows satisfy $B[j,:] = \lambda_j \Phi[p,:]$, where $\lambda_j \neq 0$. For unique identification, assume nonzero, pairwise nonproportional rows of Φ. Use the row matching procedure from Section 3.2, retaining negative scales and excluding zero scales. A negative scale corrects the corresponding signature orientation. Zero rows or ambiguous matches require additional probes; persistent ambiguities remain unresolved. Under these conditions, exact response matching recovers L. The entries of L determine the permutation, signs, and scales. Applying it to the candidate affine parameters yields

$$\widetilde{W}_h^{(1)} = L\widehat{W}_h^{(1)} = S_1 W_h^{(1)} S_1^{-1}, \qquad \widetilde{b}^{(1)} = L\widehat{b}^{(1)} = S_1 b^{(1)}.$$

4.3 Recurrent Parameter Recovery

After applying RPA, we recover subsequent current-branch layers sequentially, obtaining each $\widehat{W}_x^{(\ell)}$ once the preceding layers are in the reference coordinates. If applied without modification, conventional signature recovery treats both weight blocks as unknown. This increases the number of unknown coefficients per neuron, complicates clustering, and leaves sign recovery and alignment of neuron ordering and scaling unresolved. Therefore, we propose RPR. For each neuron, RPR uses the corresponding row of $\widehat{W}_x^{(\ell)}$ as an anchor. Only its recurrent row and bias remain unknown. A nonzero anchor fixes the common sign and scale and guides the clustering of critical or dual points.

RPR with Hard Labels. Inspired by the IsConsistent test [6] and graph clustering [7], we construct an overlap graph of local recurrent candidates. For

each known anchor, the graph selects an initial cluster of dual points, called the core.

Let N denote the number of dual point records, each comprising a dual point and its associated features and local geometry. Let a_j^T be row j of $\widehat{W}_x^{(\ell)}$. Let $w_j^\mathsf{T} = \widetilde{W}_h^{(\ell)}[j,:]$ and $b_j = \widetilde{b}^{(\ell)}[j]$ be the unknown aligned parameters. Here, $a_j \in \mathbb{R}^{d_{\ell-1}}$, $w_j \in \mathbb{R}^{d_\ell}$, and $b_j \in \mathbb{R}$. Define

$$z(x) = \left[\widetilde{v}_{\ell-1}(x)^\mathsf{T}\ \widetilde{u}_\ell(x)^\mathsf{T}\right]^\mathsf{T}, \qquad \psi_j(z) = a_j^\mathsf{T}\widetilde{v}_{\ell-1} + w_j^\mathsf{T}\widetilde{u}_\ell + b_j.$$

The critical hyperplane has normal $\theta_j = [a_j^\mathsf{T}\ w_j^\mathsf{T}]^\mathsf{T}$. Both $z(x)$ and θ_j belong to $\mathbb{R}^{d_{\ell-1}+d_\ell}$. The normal θ_j excludes the bias b_j. For $a_j \neq \mathbf{0}$, the anchor fixes the selected neuron's sign and scale. The remaining unknowns are w_j and b_j.

We briefly recall the IsConsistent procedure [6]. For dual point x_i^D, let $z_i = z(x_i^\mathrm{D})$ and $J_i = \partial z(x)/\partial x|_{x=x_i^\mathrm{D}}$. Here, $x = x^{\tau-1}$ is the variable input of $M_2^{(n)}$; all other inputs are zero vectors. The Jacobian is computed from the recovered prefix within a fixed activation region. The local dual space is the intersection of two adjacent decision boundary hyperplanes [6]. At a dual point, we estimate their normals from label queries and orthonormalize them to form $N_i \in \mathbb{R}^{d_0\times 2}$, giving the projector $\mathcal{P}_i = I_{d_0} - N_i N_i^\mathsf{T}$ onto the dual space directions. Inspired by the covariance-based formulation in [29], we define the following trace-normalized Gram matrix:

$$U_i = J_i\mathcal{P}_i, \qquad G_i = \frac{U_iU_i^\mathsf{T}}{\mathrm{tr}(U_iU_i^\mathsf{T})}.$$

Here, $J_i, U_i \in \mathbb{R}^{(d_{\ell-1}+d_\ell)\times d_0}$ and $\mathcal{P}_i \in \mathbb{R}^{d_0\times d_0}$. The columns of U_i span the image of this direction space under the feature Jacobian J_i. The normalization requires $\mathrm{tr}(U_iU_i^\mathsf{T}) > 0$. In this Gram matrix form, the consistency tests of [6,29] cluster dual points using the rank of $G_i + G_k$ over active coordinates; a one-dimensional nullspace of $\sum_{i\in\mathcal{C}} G_i$ gives the neuron signature up to scale.

The original IsConsistent test does not use known anchors to fix part of the normal vector. RPR uses this additional information to fit local recurrent weight candidates and compares their values on shared active coordinates to cluster dual points generated by the same neuron. If the dual point in record i is generated by neuron j, then $\theta_j^\mathsf{T} G_i \theta_j = 0$ in exact arithmetic. Partition the Gram matrix as

$$G_i = \begin{bmatrix} K_i & C_i^\mathsf{T} \\ C_i & H_i \end{bmatrix}.$$

The block dimensions are $K_i \in \mathbb{R}^{d_{\ell-1}\times d_{\ell-1}}$, $C_i \in \mathbb{R}^{d_\ell\times d_{\ell-1}}$, and $H_i \in \mathbb{R}^{d_\ell\times d_\ell}$. Define

$$q_{ij}(w) = a_j^\mathsf{T} K_i a_j + 2w^\mathsf{T} C_i a_j + w^\mathsf{T} H_i w.$$

The normalized dual space residual is

$$r_{ij}(w) = \frac{\sqrt{q_{ij}(w)}}{\|[a_j^\mathsf{T}\ w^\mathsf{T}]^\mathsf{T}\|_2}. \tag{13}$$

This normalization removes the scale of the concatenated normal, which excludes the bias.

We first construct an overlap graph from local recurrent candidates. Minimizing $q_{ij}(w)$ gives the local candidate with minimum norm: $w_{ij}^{\mathrm{loc}} = -H_i^{\dagger} C_i a_j$. We use $r_{ij}(w)$ for validation and define the following relative residual for local candidate screening. Each retained record forms a vertex in the overlap graph for anchor j, with associated candidate w_{ij}^{loc}. We retain record i only when

$$\delta_{ij}^{\mathrm{loc}} = \sqrt{\frac{q_{ij}(w_{ij}^{\mathrm{loc}})}{\max\{a_j^{\mathsf{T}} K_i a_j, \eta\}}} \leq \varepsilon_{\mathrm{loc}},$$

where $\eta > 0$ is a numerical floor. Let $\mathcal{A}_i$ contain the active recurrent input coordinates at x_i^{D}. For two retained records i and k, define $\mathcal{O}_{ik} = \mathcal{A}_i \cap \mathcal{A}_k$, and let P_{ik} select these common coordinates. Their overlap discrepancy is

$$D_{ik}^{(j)} = \frac{\|P_{ik}(w_{ij}^{\mathrm{loc}} - w_{kj}^{\mathrm{loc}})\|_2}{\max\left\{\frac{1}{2}\left(\|P_{ik} w_{ij}^{\mathrm{loc}}\|_2 + \|P_{ik} w_{kj}^{\mathrm{loc}}\|_2\right), \eta\right\}}.$$

Overlap agreement can suggest a candidate core.

Candidate edges must satisfy $|\mathcal{O}_{ik}| \geq m_{\mathrm{ov}}$ and $D_{ik}^{(j)} \leq \varepsilon_{\mathrm{ov}}$. The overlap test compares candidate coefficients on their shared active coordinates. It does not require either candidate to determine the complete recurrent vector. The largest connected component is denoted by $\mathcal{C}_j^{\mathrm{core}}$. It provides the initial core, subject to the validation tests below.

We then fit a common recurrent vector. For a set $\mathcal{C}$ of record indices, the aggregate dual space objective is $\sum_{i \in \mathcal{C}} q_{ij}(w)$. Its minimizer with minimum norm is

$$w(\mathcal{C}) = -\left(\sum_{i \in \mathcal{C}} H_i\right)^{\dagger} \left(\sum_{i \in \mathcal{C}} C_i a_j\right).$$

A derivation is given in Appendix A.2.

Before testing candidate merges, we estimate the bias. For a fitted vector w, let $\theta_j(w) = [a_j^{\mathsf{T}} \; w^{\mathsf{T}}]^{\mathsf{T}}$. The initial bias estimate is

$$b(\mathcal{C}) = \operatorname*{median}_{i \in \mathcal{C}} \{-z_i^{\mathsf{T}} \theta_j(w)\}.$$

We refine this estimate by median filtering. The normalized dual point residual evaluated at x_i^{D}, rather than the Euclidean distance from a point to a hyperplane, is

$$\mu_{ij}(w, b) = \frac{|z_i^{\mathsf{T}} \theta_j(w) + b|}{\|[\theta_j(w)^{\mathsf{T}} \; b]^{\mathsf{T}}\|_2}.$$

The aggregate matrix must reach numerical rank d_ℓ. For a rank-deficient core $\mathcal{C}$, refit w, b on $\mathcal{C} \cup \mathcal{D}$ for each candidate component $\mathcal{D}$. Accept the merge only if rank increases and the prescribed upper quantiles of μ_{ij} on each set separately

and r_{ij} on their union meet their respective thresholds. Repeat until full rank or the repair limit.

Next, expand the core to $\mathcal{C}_j^{\mathrm{cert}}$ using the largest adjacent ratio among sorted point residuals, with a positive numerical floor and the configured support-size and residual limits. Include records below the selected gap and always retain the core.

For validation, we partition the records indexed by $\mathcal{C}_j^{\mathrm{cert}}$ into two disjoint groups $\mathcal{S}_A$ and $\mathcal{S}_B$. Each group must contain sufficient records. Separate fits give $p_A = [w_A^\mathsf{T}\ b_A]^\mathsf{T}$ and $p_B = [w_B^\mathsf{T}\ b_B]^\mathsf{T}$. For $p_A, p_B \neq 0$, the cluster is accepted only if both aggregate systems have rank d_ℓ and

$$\frac{p_A^\mathsf{T} p_B}{\|p_A\|_2 \|p_B\|_2} \geq \gamma, \qquad \frac{\|p_A - p_B\|_2}{\frac{1}{2}(\|p_A\|_2 + \|p_B\|_2)} \leq \varepsilon_{\mathrm{par}}.$$

Each fit must also satisfy the same quantile thresholds for both residuals on the opposite group. Acceptance requires all rank, residual, and split consistency tests. An empty core, persistent rank deficiency, or failed validation leaves the neuron unresolved. We may then acquire additional points or apply FBD (Section 4.4).

Complexity. Appendix A.3 bounds the offline arithmetic cost of this procedure, including Gram matrix construction, given the records and feature Jacobians; for fixed layer widths, the pairwise overlap comparisons are quadratic in the number of retained candidates, which local residual screening reduces.

RPR with Raw Outputs. With raw outputs, gradient jumps reveal normal directions of local critical hyperplanes in input space. We can therefore combine value and gradient residuals to cluster points using the known anchors. We retain the definitions of a_j, w_j, b_j from the hard label variant.

For a candidate critical point x_i, define the recovered feature Jacobians within its fixed prefix activation region:

$$J_{u,i} := \left.\frac{\partial \widetilde{u}_\ell}{\partial x}\right|_{x=x_i} \in \mathbb{R}^{d_\ell \times d_0}, \qquad J_{v,i} := \left.\frac{\partial \widetilde{v}_{\ell-1}}{\partial x}\right|_{x=x_i} \in \mathbb{R}^{d_{\ell-1} \times d_0}.$$

This convention agrees with $J_i = [J_{v,i}^\mathsf{T}\ J_{u,i}^\mathsf{T}]^\mathsf{T}$. For differentiable inputs, let

$$g(x) := \nabla_x M_2^{(n)}(x) \in \mathbb{R}^{d_0}, \qquad \Delta g_i := g(x_i^+) - g(x_i^-).$$

Here, x_i^+ and x_i^- lie on opposite sides of the candidate critical hyperplane. Only the generating neuron changes activation status between them; the activation statuses of all other neurons remain fixed. Because the activation statuses of all other neurons remain fixed, paths that bypass the target neuron contribute identical gradients on both sides and cancel in Δg_i. Under exact recovery, the chain rule gives

$$\nabla_x [\psi_j(z(x))]|_{x=x_i} = J_i^\mathsf{T} \theta_j = J_{v,i}^\mathsf{T} a_j + J_{u,i}^\mathsf{T} w_j, \qquad \Delta g_i = c_i J_i^\mathsf{T} \theta_j.$$

The scalar c_i includes the downstream output coefficient and the crossing orientation. We retain only nonzero jumps. Finite differences of raw outputs estimate the gradients on opposite sides. Below, Δg_i denotes their measured difference. A gradient jump provides a direction but not its neuron identity. We therefore score its compatibility with each anchor. For $\Delta g_i \neq \mathbf{0}$, the score for anchor a_j is

$$s_{ij} = \min_{w,\alpha} \frac{\left\| J_{u,i}^{\mathsf{T}} w + J_{v,i}^{\mathsf{T}} a_j - \alpha \nu_i \right\|_2}{\max\{\|J_{v,i}^{\mathsf{T}} a_j\|_2, \eta\}}, \qquad \nu_i = \frac{\Delta g_i}{\|\Delta g_i\|_2}.$$

Here, $\nu_i \in \mathbb{R}^{d_0}$ is the unit direction of the gradient jump, and $\alpha \in \mathbb{R}$ is a fitted scale. The constant $\eta > 0$ prevents unstable normalization when the anchored Jacobian contribution is small. We write $\widetilde{u}_{\ell,i} = \widetilde{u}_\ell(x_i)$ and $\widetilde{v}_{\ell-1,i} = \widetilde{v}_{\ell-1}(x_i)$. We assign a seed only when its smallest score passes the configured residual threshold. The gap to the second smallest score must also meet its threshold. Otherwise, the point remains unassigned. The accepted points form $\mathcal{C}_j^{\text{seed}}$. An empty seed cluster leaves the neuron unresolved. For a nonempty cluster, initialize the critical hyperplane with its anchor fixed:

$$(w_j^{(0)}, b_j^{(0)}) = \arg\min_{w,b} \sum_{i \in \mathcal{C}_j^{\text{seed}}} \left(\widetilde{u}_{\ell,i}^{\mathsf{T}} w + \widetilde{v}_{\ell-1,i}^{\mathsf{T}} a_j + b \right)^2.$$

Starting from these initial critical hyperplanes, we alternate point reassignment and parameter fitting to refine the clusters and recovered parameters. Let T_{iter} denote the number of iterations. For a provisional critical hyperplane (a_j, w_j, b_j), reassignment uses the following distance in feature space and gradient jump compatibility score:

$$d_{ij} = \frac{\left| \widetilde{u}_{\ell,i}^{\mathsf{T}} w_j + \widetilde{v}_{\ell-1,i}^{\mathsf{T}} a_j + b_j \right|}{\sqrt{\|w_j\|_2^2 + \|a_j\|_2^2}}, \qquad \chi_{ij} = \frac{\left| \left(J_{u,i}^{\mathsf{T}} w_j + J_{v,i}^{\mathsf{T}} a_j \right)^{\mathsf{T}} \Delta g_i \right|}{\|J_{u,i}^{\mathsf{T}} w_j + J_{v,i}^{\mathsf{T}} a_j\|_2 \|\Delta g_i\|_2}.$$

The cosine score requires $J_{u,i}^{\mathsf{T}} w_j + J_{v,i}^{\mathsf{T}} a_j \neq \mathbf{0}$. Point i is assigned to neuron j only when

$$j = \arg\min_k d_{ik} = \arg\max_k \chi_{ik}$$

with unique extrema and with the corresponding thresholds satisfied. Otherwise, it remains unassigned. Reassignment stops when assignments stabilize or the configured iteration budget is exhausted. Neither stopping condition alone certifies recovery. For rank completion, test unassigned points for rank gain and consistency with the fitted hyperplane and gradient jump, then refit after each accepted point (Appendix A.3).

For final recovery, we use a hard-anchor fit: the recovered vector a_j remains fixed. Only the recurrent vector w_j and bias b_j are optimized:

$$\min_{w,b} \sum_{i \in \mathcal{C}_j} \left(\widetilde{u}_{\ell,i}^{\mathsf{T}} w + \widetilde{v}_{\ell-1,i}^{\mathsf{T}} a_j + b \right)^2.$$

Construct $A_j \in \mathbb{R}^{|\mathcal{C}_j| \times (d_\ell + 1)}$ and $y_j \in \mathbb{R}^{|\mathcal{C}_j|}$ with rows and entries

$$A_j[i,:] = [\widetilde{u}_{\ell,i}^{\mathsf{T}} \;\; 1], \qquad y_j[i] = -\widetilde{v}_{\ell-1,i}^{\mathsf{T}} a_j, \quad i \in \mathcal{C}_j.$$

Here, rows are indexed by the records in $\mathcal{C}_j$. The least squares estimate is

$$\begin{bmatrix} w_j \\ b_j \end{bmatrix} = A_j^{\dagger} y_j.$$

Unique recovery requires $\operatorname{rank}(A_j) = d_\ell + 1$. Accept the fitted row only after the rank and configured residual checks pass. Otherwise, acquire additional records or apply FBD; unsuccessful recovery remains unresolved. Accepted estimates give $\widetilde{W}_h^{(\ell)}[j,:] = w_j^{\mathsf{T}}$ and $\widetilde{b}^{(\ell)}[j] = b_j$.

Complexity. Appendix A.3 bounds the offline arithmetic cost of initialization, iterative clustering, rank completion, and final recovery given the records with their features, Jacobians, and gradient jumps; for fixed layer widths and a bounded number of iterations and candidate checks, this cost is linear in the number of records.

4.4 Folded Bias Difference

RPR labels a neuron difficult when it cannot obtain a full-rank fit that passes validation, due to sparse boundary observations, limited activation diversity, rank deficiency, or unstable fits across data splits. For the hard-label 200-32^3-1 target, FBD recovers two difficult neurons in $M_2^{(3)}$, while RPR's overlap procedure recovers the remaining 31 neurons in each layer.

We use the models $M_{\xi_k}^{(n)}(x)$ from Fig. 9. Each probe $\xi_k \sim \mathcal{N}(\mathbf{0}, I_{d_0})$ fixes the penultimate input; the final input varies. Equation (12) defines $\alpha_k^{(1)}$ and $r_k^{(1)}$. For $2 \le \ell \le n$,

$$\alpha_k^{(\ell)} = W_x^{(\ell)} \sigma\left(\alpha_k^{(\ell-1)}\right) + \beta_{\tau-1}^{(\ell)}, \qquad r_k^{(\ell)} = W_h^{(\ell)} \sigma\left(\alpha_k^{(\ell)}\right) + b^{(\ell)}.$$

Recovery at layer ℓ assumes aligned recurrent parameters for all preceding current branch layers. Let $\widetilde{z}_{\ell-1,k}(x)$ denote the recovered prefix output of $M_{\xi_k}^{(n)}$. For neuron j, acquire an associated critical point or dual point $x_{j,k}^*$. Its zero preactivation gives the folded bias observation

$$\widehat{r}_k^{(\ell)}[j] = -\widehat{W}_x^{(\ell)}[j,:]\, \widetilde{z}_{\ell-1,k}(x_{j,k}^*).$$

Thus, this observation uses known input weights, the recovered prefix, and a queried point. It does not require the unknown recurrent row. Under exact recovery, $\widehat{r}_k^{(\ell)} = S_\ell r_k^{(\ell)}$.

The recovered previous branch gives $\widehat{\alpha}_k^{(\ell)} = S_\ell \alpha_k^{(\ell)}$. Each retained probe provides the following linear equation in the unknown recurrent row and bias:

$$\widetilde{W}_h^{(\ell)}[j,:]\, \sigma\left(\widehat{\alpha}_k^{(\ell)}\right) + \widetilde{b}^{(\ell)}[j] = \widehat{r}_k^{(\ell)}[j].$$

Both observed biases and computed activations use the coordinates fixed by S_ℓ. A unique exact solution therefore satisfies Equation (10), without further sign recovery or alignment.

Choose one retained probe as the reference and relabel its index as 0. Subtracting its equation eliminates the bias:

$$\widetilde{W}_h^{(\ell)}[j,:]\left(\sigma\left(\widehat{\alpha}_k^{(\ell)}\right)-\sigma\left(\widehat{\alpha}_0^{(\ell)}\right)\right)=\widehat{r}_k^{(\ell)}[j]-\widehat{r}_0^{(\ell)}[j].$$

Stack the M retained differences as

$$\Delta\mathsf{H}\left(\widetilde{W}_h^{(\ell)}[j,:]\right)^{\mathsf{T}}=\Delta\boldsymbol{r}_j,\qquad \Delta\mathsf{H}\in\mathbb{R}^{M\times d_\ell},\quad \Delta\boldsymbol{r}_j\in\mathbb{R}^M.$$

Unique recovery requires $\operatorname{rank}(\Delta\mathsf{H})=d_\ell$. In practice, least squares estimates the recurrent row. The original equations then estimate the bias:

$$\widetilde{b}_{j,\mathrm{est}}^{(\ell)}=\operatorname{median}_k\left(\widehat{r}_k^{(\ell)}[j]-\widetilde{W}_{h,\mathrm{est}}^{(\ell)}[j,:]\,\sigma\left(\widehat{\alpha}_k^{(\ell)}\right)\right).$$

FBD reduces the maximum prefix depth from the variable input to the target neuron. This depth is $\ell+1$ in $M_2^{(n)}$, but ℓ in $M_{\xi_k}^{(n)}$. The full model still contains n hidden layers. FBD still requires sufficient activation variation across probes to obtain full rank. Each probe also requires new critical or dual points, increasing query cost. We therefore use RPR for most neurons and reserve FBD for difficult cases.

5 Experiments

For simplicity, we use binary classifiers derived from the UEA InsectWingbeat dataset [3], mapping "Fruit flies" and "House flies" to binary labels. Each observation is a sequence of wingbeat spectrum vectors, and the model predicts its class from their temporal evolution; the attacker queries the model directly in continuous input space. Our recurrent-parameter recovery and alignment procedures do not use or modify output-layer parameters, so the reductions and hidden-state alignment extend naturally to multiclass models with affine-logit outputs.

Using the official equal-length representation, each input has $\tau = 20$ time steps and $d_0 = 200$ features ($\mathbb{R}^{20\times 200}$), requiring no resampling. Training details appear in Appendix A.4.

Table 1 reports five architectures in both observation settings; the ten baseline experiments span one to three recurrent layers. In the hard-label setting, the attack accesses only predicted labels; target logits are reserved for subsequent output-error evaluation, while target parameters are used solely by the two-layer verifier for neuron permutation and scale alignment.

For real-data fidelity, we additionally evaluate the ten baseline recovered models on the InsectWingbeat training and test partitions, each containing 5,000

sequences. Table 2 in Appendix A.5 reports error metrics, output-label agreement, and runtimes for each partition, and all five hard-label models achieve 100% agreement with their target labels on both partitions.

For longer sequences, the corresponding results are reported in Table 1; runtime and related details appear in Appendix A.6. On 10^5 complete standard Gaussian sequences, the maximum absolute output error is $2^{-23.30}$, comparable to the $\tau = 20$ result in Table 1.

6 Discussion and Conclusions

Discussion. Our attack targets many-to-one ReLU RNNs with independently controllable continuous inputs. For many-to-many RNNs with the same recurrence and query access, our reductions apply without using intermediate outputs. For one-to-many RNNs that receive input only at the first step, RPA is not directly applicable because it requires independent control of two consecutive inputs. Gated architectures are not covered either. LSTMs [15] and GRUs [11] share their parameters across time but combine interacting affine transformations with nonlinear gates, so our reductions do not directly apply, and extending them to gated architectures and other activations remains an open direction.

We do not study concrete defenses. Since our reductions use controlled input families, a defender could reject sequences outside the legitimate distribution, although whether plausible inputs can induce the required activation patterns remains open. In the raw-output setting, rounding, quantization, or noise may hinder FCN signature recovery, whereas the hard-label attack receives unchanged information as long as output processing preserves the returned labels.

Conclusions. We controlled inputs to construct equivalent feedforward models whose depth is independent of the sequence length, and we resolved the incompatible neuron orderings and scales that independent recovery leaves behind. RPA establishes recurrent consistency by matching folded biases across controlled probes and, for stacked RNNs, also recovers the signs of the first recurrent layer. RPR uses the known input weights as anchors to recover the deeper recurrent weight rows and biases, and FBD supplements it for difficult neurons using differences between recovered folded biases. Ten baseline extraction experiments cover five architectures in both observation settings, with one additional longer-sequence raw-output rerun at $\tau = 1024$, and under our access assumptions, consistent ordering and scaling make locally recovered parameters reusable across time.

References

1. Aad, G., et al.: Artificial neural networks on FPGAs for real time energy reconstruction of the ATLAS liquid argon calorimeters. Computing and Software for Big Science **5**(1), 19 (2021). https://doi.org/10.1007/s41781-021-00066-y

2. Asselineau, R., Derbez, P., Fouque, P.A., Minaud, B.: Cryptanalytic extraction of deep neural networks with non-linear activations. In: Heninger, N., Rosulek, M. (eds.) Advances in Cryptology – CRYPTO 2026. pp. 36–66. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-032-35415-0_2
3. Bagnall, A., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.: The UEA multivariate time series classification archive, 2018. CoRR **abs/1811.00075** (2018). https://doi.org/10.48550/arXiv.1811.00075, https://arxiv.org/abs/1811.00075
4. Canales-Martínez, I.A., Chávez-Saab, J., Hambitzer, A., Rodríguez-Henríquez, F., Satpute, N., Shamir, A.: Polynomial time cryptanalytic extraction of neural network models. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024. pp. 3–33. Springer Nature Switzerland, Cham (2024)
5. Canales-Martínez, I.A., Santos, D.: Extracting some layers of deep neural networks in the hard-label setting. In: Escudero, D., Damgård, I. (eds.) Progress in Cryptology - LATINCRYPT 2025 - 9th International Conference on Cryptology and Information Security in Latin America, Medellín, Colombia, October 1-3, 2025, Proceedings. pp. 399–421. Lecture Notes in Computer Science, Springer (2025). https://doi.org/10.1007/978-3-032-06754-8_15, https://doi.org/10.1007/978-3-032-06754-8_15
6. 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.A. (eds.) Advances in Cryptology – EUROCRYPT 2025. pp. 364–396. Springer Nature Switzerland, Cham (2025), extended version: https://eprint.iacr.org/2024/1580
7. Carlini, N., Jagielski, M., Mironov, I.: Cryptanalytic extraction of neural network models. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 189–218. Springer International Publishing, Cham (2020)
8. Chen, Y., Dong, X., Guo, J., Shen, Y., Wang, A., Wang, X.: Hard-label cryptanalytic extraction of neural network models. In: Chung, K.M., Sasaki, Y. (eds.) Advances in Cryptology – ASIACRYPT 2024. pp. 207–236. Springer Nature Singapore, Singapore (2025)
9. Chen, Y., Dong, X., Ma, R., Shen, Y., Wang, A., Yu, H., Wang, X.: Delving into cryptanalytic extraction of PReLU neural networks. In: Hanaoka, G., Yang, B.Y. (eds.) Advances in Cryptology – ASIACRYPT 2025. pp. 576–607. Springer Nature Singapore, Singapore (2025). https://doi.org/10.1007/978-981-95-5096-8_18
10. Chen, Z., Tang, S., Gao, Z., Su, Y., Qin, L., Dong, X.: Algebraic attack on convolutional neural networks with max pooling. In: Advances in Cryptology – CRYPTO 2026. pp. 3–35. Springer (2026)
11. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics (2014). https://doi.org/10.3115/v1/D14-1179
12. Coqueret, B., Carbone, M., Sentieys, O., Zaid, G.: A divide-and-conquer strategy for hard-label extraction of deep neural networks via side-channel attacks. Cryptology ePrint Archive, Paper 2024/1870 (2024), https://eprint.iacr.org/2024/1870
13. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990), https://doi.org/10.1207/s15516709cog1402_1
14. 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 37: 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
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
16. 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. pp. 67–98. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-032-35415-0_3
17. Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., Papernot, N.: High accuracy and high fidelity extraction of neural networks. In: 29th USENIX Security Symposium (USENIX Security 20). pp. 1345–1362. USENIX Association (2020), https://www.usenix.org/conference/usenixsecurity20/presentation/jagielski
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
19. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1–2), 83–97 (1955). https://doi.org/10.1002/nav.3800020109
20. Lalapura, V.S., Bhimavarapu, V.R., Amudha, J., Satheesh, H.S.: A systematic evaluation of recurrent neural network models for edge intelligence and human activity recognition applications. Algorithms **17**(3), 104 (2024). https://doi.org/10.3390/a17030104
21. Le, Q.V., Jaitly, N., Hinton, G.E.: A simple way to initialize recurrent networks of rectified linear units. arXiv preprint arXiv:1504.00941 (2015). https://doi.org/10.48550/arXiv.1504.00941
22. Liu, H., Siproudhis, A., Boura, C., Peyrin, T.: Model extraction of convolutional neural networks with max-pooling. Cryptology ePrint Archive, Paper 2026/464 (2026), https://eprint.iacr.org/2026/464
23. 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. pp. 482–512. Springer Nature Switzerland, Cham (2026)
24. Ma, R., Chen, Y., Zhang, J., Yu, H., Wang, X.: Breaking slope and structure restrictions: Broadening hard-label cryptanalytic extraction of PReLU neural networks. Cryptology ePrint Archive, Paper 2026/1066 (2026), https://eprint.iacr.org/2026/1066
25. Martinez, M., Paparella, P.: The group of monomial matrices. Mathematics Exchange **17**(1), 34–38 (Mar 2026). https://doi.org/10.33043/94EV9NDNHD, https://openjournals.bsu.edu/mathexchange/article/view/6158
26. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel. pp. 807–814. Omnipress (2010), https://icml.cc/Conferences/2010/papers/432.pdf
27. Qi, X., Lei, H., Wei, L., Sun, X., Wang, M.: Cryptanalytic extraction of neural networks with various activation functions. IACR Trans. Symmetric Cryptol. **2026**(1), 468–505 (2026). https://doi.org/10.46586/TOSC.V2026.I1.468-505

28. Rolnick, D., Kording, K.: Reverse-engineering deep ReLU networks. In: Daumé III, H., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 8178–8187. PMLR (2020), https://proceedings.mlr.press/v119/rolnick20a.html
29. Sun, X., Lei, H., Wei, L., Qi, X., Hu, K., Wang, M., Wang, W.: Cryptanalytic extraction of convolutional neural networks. In: Ma, S., Nepal, S. (eds.) Information Security and Privacy – ACISP 2026. pp. 123–147. Springer Nature Singapore, Singapore (2026). https://doi.org/10.1007/978-981-92-3012-9_6
30. Takemura, T., Yanai, N., Fujiwara, T.: Model extraction attacks on recurrent neural networks. Journal of Information Processing **28**, 1010–1024 (2020). https://doi.org/10.2197/ipsjjip.28.1010, https://www.jstage.jst.go.jp/article/ipsjjip/28/0/28_1010/_article/-char/en
31. Tran, V.H., Vo, T., Tran-Huu, T., Nguyen, A.T., Nguyen, T.: Monomial matrix group equivariant neural functional networks. Advances in Neural Information Processing Systems **37**, 48628–48665 (2024)
32. Wang, S., Lei, H., Wei, L., Liu, Q., Hu, K., Wang, M.: Cryptanalytic extraction of multi-head softmax attention models. Cryptology ePrint Archive, Paper 2026/1678 (2026), https://eprint.iacr.org/2026/1678

A Supporting Proofs and Analysis

A.1 Proof of Theorem 2

Proof. Let $S_0 = I_{d_0}$ and $h_t^{(0)} = \widetilde{h}_t^{(0)} = x^t$. The initial states satisfy $\widetilde{h}_0^{(\ell)} = S_\ell h_0^{(\ell)} = \mathbf{0}$. Fix $1 \le t \le \tau$. Assume the hidden state relation holds at time $t-1$ for every layer. At time t, proceed through layers $1, \dots, n$. For layer ℓ, assume the relation holds for layer $\ell - 1$. The parameter relations in Equations (8)–(10) and $\sigma(S_\ell r) = S_\ell \sigma(r)$ give

$$\begin{aligned}\widetilde{h}_t^{(\ell)} &= \sigma\Big(S_\ell W_x^{(\ell)} S_{\ell-1}^{-1} \widetilde{h}_t^{(\ell-1)} + S_\ell W_h^{(\ell)} S_\ell^{-1} \widetilde{h}_{t-1}^{(\ell)} + S_\ell b^{(\ell)}\Big) \\ &= S_\ell \sigma\Big(W_x^{(\ell)} h_t^{(\ell-1)} + W_h^{(\ell)} h_{t-1}^{(\ell)} + b^{(\ell)}\Big) = S_\ell h_t^{(\ell)}.\end{aligned}$$

The input identity starts the induction over layers. This proves the relation at time t for every recurrent layer. Induction over time then establishes the result for all $1 \le t \le \tau$. Finally,

$$\widetilde{f}(X) = \widehat{W}_o \widetilde{h}_\tau^{(n)} + \widehat{b}_o = c W_o S_n^{-1} S_n h_\tau^{(n)} + c b_o = c f(X).$$

The positivity of c proves Equation (11).

A.2 Minimizer of the Anchored Aggregate Objective

Lemma 1 (Closed form of the aggregate minimizer). *Fix a_j. For each $i \in \mathcal{C}$, partition the Gram matrix G_i into K_i, C_i, H_i. Then every minimizer of*

$$\mathcal{Q}_j(w) = \sum_{i \in \mathcal{C}} \left(a_j^\mathsf{T} K_i a_j + 2 w^\mathsf{T} C_i a_j + w^\mathsf{T} H_i w\right)$$

satisfies the normal equation $\left(\sum_{i\in\mathcal{C}} H_i\right) w = -\sum_{i\in\mathcal{C}} C_i a_j$. *The solution with minimum norm is* $w(\mathcal{C}) = -(\sum_{i\in\mathcal{C}} H_i)^\dagger \sum_{i\in\mathcal{C}} C_i a_j$, *including when the aggregate matrix is singular. If* $\sum_{i\in\mathcal{C}} H_i$ *is invertible, Equation* (14) *gives the unique minimizer.*

Proof. Each Gram matrix G_i is positive semidefinite. Its principal submatrix H_i is also positive semidefinite. Thus, $\mathcal{Q}_j$ is a convex quadratic in w. Its minimizers are exactly its stationary points. Differentiating gives

$$\nabla_w \mathcal{Q}_j(w) = 2\sum_{i\in\mathcal{C}} \left(C_i a_j + H_i w\right).$$

Thus, the stationarity condition $\nabla_w \mathcal{Q}_j(w) = 0$ yields

$$\left(\sum_{i\in\mathcal{C}} H_i\right) w = -\sum_{i\in\mathcal{C}} C_i a_j.$$

For the singular case, abbreviate the aggregate matrix as $H = \sum_{i\in\mathcal{C}} H_i$. Let $\boldsymbol{g}_\mathcal{C} = \sum_{i\in\mathcal{C}} C_i a_j$. For any $v \in \ker H$, nonnegativity of the Gram quadratic gives $\mathcal{Q}_j(tv) = \mathcal{Q}_j(0) + 2tv^\mathsf{T}\boldsymbol{g}_\mathcal{C} \geq 0$ for every real t. Hence $v^\mathsf{T}\boldsymbol{g}_\mathcal{C} = 0$, so $\boldsymbol{g}_\mathcal{C} \in \operatorname{range}(H)$. The normal equation is therefore consistent, and its solution with minimum norm is $-H^\dagger \boldsymbol{g}_\mathcal{C}$.

The constant term $\sum_{i\in\mathcal{C}} a_j^\mathsf{T} K_i a_j$ does not affect this derivative. The aggregate objective has a unique minimizer if and only if $\operatorname{rank}(H) = d_\ell$. Then $\sum_{i\in\mathcal{C}} H_i$ is positive definite and invertible. The unique minimizer is then

$$w(\mathcal{C}) = -\left(\sum_{i\in\mathcal{C}} H_i\right)^{-1} \left(\sum_{i\in\mathcal{C}} C_i a_j\right). \tag{14}$$

A.3 Complexity

Hard labels. We analyze the hard label procedure in Section 4.3, given the dual point records and their feature Jacobians, including Gram matrix construction. Let N be the number of records and $n_a = d_\ell$ the number of anchors. For anchor j, let $N_j \leq N$ count records retained after local residual screening and q_j count additional connected components examined for rank repair. For this bound, assume $d_{\ell-1} = O(d_\ell)$ and a bounded number of median filtering iterations.

Let $m = d_{\ell-1} + d_\ell$. Using $U_i = J_i - (J_i N_i)N_i^\mathsf{T}$ with $N_i \in \mathbb{R}^{d_0\times 2}$, the projection costs $O(md_0)$ per record without forming a dense projector. Computing $U_i U_i^\mathsf{T}$ costs $O(m^2 d_0)$, and trace normalization costs $O(m^2)$. Thus, Gram matrix construction costs $O(Nd_0 d_\ell^2)$ in total. This count excludes dual point acquisition and feature Jacobian evaluation.

The pseudoinverse of the Gram matrix block $H_i \in \mathbb{R}^{d_\ell\times d_\ell}$ can be computed by eigendecomposition in $O(d_\ell^3)$ time. Since H_i is independent of the anchor, this decomposition is performed once per record and reused, costing $O(Nd_\ell^3)$ in total. This decomposition cost is additional to Gram matrix construction.

Computing all local candidates w_{ij}^{loc} and their screening residuals $\delta_{ij}^{\mathrm{loc}}$ costs $O(n_a N d_\ell^2)$. Pairwise overlap comparisons for anchor j cost $O(N_j^2 d_\ell)$; finding the largest connected component costs at most $O(N_j^2)$.

For each fit, aggregating $\sum_{i\in\mathcal{C}} H_i$ and $\sum_{i\in\mathcal{C}} C_i a_j$ costs $O(|\mathcal{C}|d_\ell^2)$. Rank checking and pseudoinverse fitting share a decomposition of the aggregate matrix, costing $O(d_\ell^3)$. Evaluating $r_{ij}(w)$ costs $O(d_\ell^2)$ per record and $O(Nd_\ell^2)$ for at most N records. Since $|\mathcal{C}| \le N$, these steps together cost $O(d_\ell^3 + Nd_\ell^2)$ per fit. Bias estimation and dual point residual evaluation are also covered by this bound.

Fitting the initial core and checking q_j additional connected components for rank repair cost $O((1+q_j)(d_\ell^3 + Nd_\ell^2))$. With a bounded number of partition trials, validation costs $O(d_\ell^3 + Nd_\ell^2)$. This additional cost is absorbed into the same $O((1+q_j)(d_\ell^3+Nd_\ell^2))$ bound. Sorting dual point residuals and locating the expansion threshold cost $O(N \log N)$ per anchor.

Combining these steps gives the bound

$$O\left(Nd_0d_\ell^2 + Nd_\ell^3 + n_aNd_\ell^2 + \sum_{j=1}^{n_a}\left[N_j^2 d_\ell + (1+q_j)(d_\ell^3 + Nd_\ell^2) + N\log N\right]\right).$$

For fixed layer widths, pairwise overlap comparisons are quadratic in the number of retained candidates. Local residual screening reduces this cost by reducing N_j.

Raw outputs. Given candidate records with their features, Jacobians, and gradient jumps, we analyze initialization, alternating point reassignment and parameter fitting, rank completion, and final recovery. Record acquisition, feature and Jacobian evaluation, and gradient jump measurement are excluded. Let N count candidate records, $n_a = d_\ell$ count anchors, and T_{iter} count global reassignment and fitting rounds, including updates after rank completion. The input dimension is d_0, following Section 2.3.

For each point and anchor, computing s_{ij} solves a least squares problem with design matrix $[J_{u,i}^\mathsf{T} \ -\nu_i]$, of size $d_0 \times (d_\ell + 1)$, and right hand side $-J_{v,i}^\mathsf{T} a_j$. A conservative cost per pair is $O(d_0 d_\ell^2 + d_\ell^3 + d_0 d_{\ell-1})$. Sorting scores for initial assignment costs $O(Nn_a \log n_a)$, and fitting all initial critical hyperplanes costs $O(Nd_\ell^2 + n_a d_\ell^3)$. Thus, we use

$$C_{\mathrm{seed}} = O\big(Nn_a(d_0d_\ell^2 + d_\ell^3 + d_0d_{\ell-1} + \log n_a) + Nd_\ell^2 + n_a d_\ell^3\big).$$

In each iteration, evaluating the distances d_{ij} requires the inner products $\widetilde{u}_{\ell,i}^\mathsf{T} w_j$ and $\widetilde{v}_{\ell-1,i}^\mathsf{T} a_j$. This costs $O(d_\ell + d_{\ell-1})$ per point and neuron pair, or $O(Nn_a(d_\ell + d_{\ell-1}))$ in total. Computing the predicted normal $J_{u,i}^\mathsf{T} w_j + J_{v,i}^\mathsf{T} a_j$ costs $O(d_0(d_\ell + d_{\ell-1}))$ per pair. Its normalized inner product with Δg_i gives χ_{ij} in an additional $O(d_0)$, so evaluating all χ_{ij} costs $O(Nn_a d_0(d_\ell + d_{\ell-1}))$.

For each point, sorting d_{ij} and χ_{ij} over the n_a neurons to select an assignment and check the thresholds costs $O(n_a \log n_a)$, or $O(Nn_a \log n_a)$ in total. With assignments fixed, updating (w_j, b_j) solves the least squares system defined by A_j and y_j. For a cluster $\mathcal{C}_j$, this costs $O(|\mathcal{C}_j|d_\ell^2 + d_\ell^3)$. Since each point is assigned

to at most one neuron, $\sum_j |\mathcal{C}_j| \le N$, giving $O(Nd_\ell^2 + n_a d_\ell^3)$ for all parameter updates. Here, the fixed terms $\widetilde{v}_{\ell-1,i}^\mathsf{T} a_j$ can be reused from distance evaluation.

For rank completion, use A_j, y_j from Section 4.3 and define the full feature matrix

$$F_j[i,:] = [\widetilde{u}_{\ell,i}^\mathsf{T}\ \widetilde{v}_{\ell-1,i}^\mathsf{T}\ 1], \qquad F_j \in \mathbb{R}^{|\mathcal{C}_j| \times (d_\ell + d_{\ell-1} + 1)}.$$

In exact arithmetic, the common critical hyperplane gives a nonzero null vector of F_j, so its rank is at most $d_\ell + d_{\ell-1}$. Write the current fit as $p_j = [w_j^\mathsf{T}\ b_j]^\mathsf{T} = A_j^\dagger y_j$. Let Z_j contain an orthonormal basis of $\ker A_j$, with first d_ℓ rows $Z_{w,j}$. For an unassigned point i, solve

$$\min_{\lambda,\alpha} \left\| J_{u,i}^\mathsf{T}(w_j + Z_{w,j}\lambda) + J_{v,i}^\mathsf{T} a_j - \alpha \nu_i \right\|_2 .$$

Normalize this minimum as in s_{ij}, and evaluate the point residual μ_{ij} using $p_j' = p_j + Z_j\lambda$. Add i only if the rank of A_j or F_j increases and both residuals meet their thresholds. Refit after each addition; stop when the ranks reach $d_\ell + 1$ and $d_\ell + d_{\ell-1}$, respectively, or no candidate passes. Full column rank of A_j ensures uniqueness of the fit with the anchor fixed; the target rank of F_j is an additional condition used in this completion procedure.

For rank completion, let R count all candidate point and cluster checks, including repeated checks after adding a point. It does not count global reassignment rounds. All records used are included in N. For this bound, assume $d_{\ell-1} = O(d_\ell)$. If a cluster contains m_j points, recomputing the ranks of the augmented A_j and F_j costs $O(m_j d_\ell^2 + d_\ell^3)$. Refitting p_j and computing Z_j are covered by the same bound. The displayed least squares problem has at most $d_\ell + 2$ unknowns, costing at most $O(d_0 d_\ell^2 + d_\ell^3)$. Thus, using $m_j \le N$, a conservative bound is

$$C_{\text{rank}} = O\big(R\left[Nd_\ell^2 + d_\ell^3 + d_0 d_\ell^2\right]\big).$$

The factor N bounds the number of rows processed in each rank check; it does not represent global point rescoring. Global reassignment and refitting after completion are counted in T_{iter}, separately from R. After iteration, final recovery solves $[w_j^\mathsf{T}\ b_j]^\mathsf{T} = A_j^\dagger y_j$ for each cluster with the anchor fixed. Summing $O(|\mathcal{C}_j| d_\ell^2 + d_\ell^3)$ over all clusters adds $O(Nd_\ell^2 + n_a d_\ell^3)$. We combine C_{seed} and C_{rank} with all T_{iter} rounds and the final fit. Using $n_a = d_\ell$ and $d_{\ell-1} = O(d_\ell)$, we absorb sorting and lower order terms. The resulting bound is

$$O\big(d_\ell^2\left[Nd_\ell(d_0 + d_\ell) + (T_{\text{iter}} + 1)(Nd_0 + d_\ell^2) + R(N + d_0 + d_\ell)\right]\big).$$

For fixed layer widths, d_0, T_{iter}, and R, this arithmetic cost bound is linear in N. If R grows with N, the rank completion term must be retained explicitly.

A.4 Training Details

We retain the official training and test partitions, reserving a stratified 20% of the training split for validation. Each classifier uses one or more ReLU recurrent layers with a linear binary-logit output and is trained with Adam [18] and binary

cross-entropy loss (learning rate 0.001, gradient-clipping threshold 1, at most 30 epochs, and minibatches of 64); we retain the checkpoint with the highest validation accuracy.

A.5 Fidelity on InsectWingbeat

We evaluate 5,000 training and 5,000 test sequences from InsectWingbeat without additional normalization or resampling. Each sequence has 20 time steps and 200 features; hidden states start at zero. This table is a separate real-data evaluation from the synthetic Gaussian evaluation reported in Table 1. Its entries were checked against the corresponding training and test evaluation logs and rounded to two decimal places as stated in the caption.

Table 2: Fidelity on the InsectWingbeat dataset. $\triangle$: maximum absolute logit error; $\diamond$: 75th-percentile relative logit error. Exponents are rounded to two decimals. Depth counts recurrent layers; agreement compares recovered and target labels.

			InsectWingbeat training data			InsectWingbeat test data			
Architecture	Depth	Setting	$(\varepsilon, 0)^{\triangle}$	$(\varepsilon, 0.25)^{\diamond}$	Agreement (%)	$(\varepsilon, 0)^{\triangle}$	$(\varepsilon, 0.25)^{\diamond}$	Agreement (%)	Runtime (s)
200-32-1	1	Raw-output	$2^{-22.06}$	$2^{-26.86}$	–	$2^{-21.98}$	$2^{-26.87}$	–	$2^{10.89}$
200-32-24-1	2	Raw-output	$2^{-20.45}$	$2^{-24.72}$	–	$2^{-20.50}$	$2^{-24.74}$	–	$2^{9.53}$
200-32^2-1	2	Raw-output	$2^{-20.00}$	$2^{-23.10}$	–	$2^{-19.24}$	$2^{-23.07}$	–	$2^{9.46}$
200-32-24-16-1	3	Raw-output	$2^{-16.66}$	$2^{-19.88}$	–	$2^{-16.65}$	$2^{-19.74}$	–	$2^{12.24}$
200-32^3-1	3	Raw-output	$2^{-10.46}$	$2^{-14.66}$	–	$2^{-10.48}$	$2^{-14.63}$	–	$2^{10.13}$
200-32-1	1	Hard-label	$2^{-18.92}$	$2^{-22.21}$	100	$2^{-19.30}$	$2^{-22.11}$	100	$2^{14.33}$
200-32-24-1	2	Hard-label	$2^{-13.22}$	$2^{-16.06}$	100	$2^{-13.57}$	$2^{-16.11}$	100	$2^{16.53}$
200-32^2-1	2	Hard-label	$2^{-14.14}$	$2^{-16.47}$	100	$2^{-14.14}$	$2^{-16.44}$	100	$2^{16.66}$
200-32-24-16-1	3	Hard-label	$2^{-12.80}$	$2^{-15.75}$	100	$2^{-12.80}$	$2^{-15.66}$	100	$2^{15.54}$
200-32^3-1	3	Hard-label	$2^{-11.41}$	$2^{-12.99}$	100	$2^{-11.27}$	$2^{-13.01}$	100	$2^{15.23}$

Agreement is the percentage of evaluated sequences for which the extracted and target models predict the same label.

A.6 Longer Sequences

The raw-output 200-32-1 experiment at $\tau = 1024$ reuses the same target weights and biases as the $\tau = 20$ baseline, without retraining. For the raw-output 200-32-1 target at $\tau = 1024$, evaluation uses 10^5 complete sequences with entries sampled independently from $\mathcal{N}(0, 1)$ at every time step. The 75th-percentile relative output error is $2^{-26.83}$. Recovery uses approximately $2^{21.58}$ queries.

The complete pipeline, including finalization, takes approximately $2^{9.44}$ s.