Toward Total Interpretability of a Tiny Elman RNN:

What We Can Show, What We Cannot Yet, and What Remains

Claude and MJC

19 February 2026

Abstract

We attempt a complete interpretation of a trained Elman RNN (128 hidden units, 82,304 parameters) that memorizes 1024 bytes of English text at 0.079 bpc. The attempt partially succeeds and reveals an unexpected decomposition.

What works: (1) an exact RNN–UM isomorphism; (2) the hidden state is a Boolean automaton (98.9% sign-determined); (3) every neuron is a 2-offset conjunction detector ($R^2 = 0.837$).

What sparse differentiation reveals: (4) the model decomposes into a **carrier regime** (86% of neurons, carrying timing) and a **pattern regime** (14%, carrying byte identity); (5) 87% of the output logit comes from the carrier (word length, tag structure), 13% from the current byte; (6) Jacobian-weighted pattern attribution shows only offsets 0 and 1 carry consistent byte-specific information; offsets 2+ average to zero across positions.

This reinterprets the factor map: offset-7 correlations are not causal byte propagation but *timing*—the byte 7 positions ago set a counter that the carrier now reads. The model is primarily a timing machine with local pattern matching, not a deep skip-pattern detector.

1 Introduction

What does it mean to fully understand a neural network? We propose a concrete answer grounded in the Universal Model (UM) [1]: you understand a model totally when you can identify the data patterns it has learned, and write those patterns back into the model's weights, recovering trained performance.

Total information of a byte is $I(E) = \log_2 |E| = 8$ bits. A model at H bpc explains $8 - H$ bits:

$$\underbrace{I(E)}_{\text{total}} = \underbrace{I(E) - H_{\text{model}}}_{\text{explained}} + \underbrace{H_{\text{model}}}_{\text{residual}} \tag{1}$$

An $X\%$ explanation captures $X\%$ of the model's explained information:

$$X\% = \frac{8 - H_{\text{explanation}}}{8 - H_{\text{trained}}} \times 100 \tag{2}$$

Our test case is deliberately small: a 128-hidden Elman RNN [2] trained on 1024 bytes of English text. The model memorizes the data (0.079 bpc after 4000 epochs). This is the point: memorization on a tiny dataset gives us a model that fails to generalize, but in a way we can study completely. What the model memorizes is a set of **skip-patterns**—conjunctions of input bytes at temporal distance—which is exactly what Elman's architecture was designed to capture.

Skip-patterns are the right mechanism for language: characters at a distance are correlated because they belong to the same word, phrase, or discourse structure. On 1024 bytes the model can only memorize which specific bytes co-occur at which offsets. With more data and hidden dimensions, the same mechanism would capture generalizable English structure. Our goal is to understand *how* the model encodes skip-patterns in its weights, and then to write those patterns back.

This paper reports what we have accomplished and what remains.

2 The Model

Standard Elman RNN [2]:

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h) \tag{3}$$

$$y_t = \mathrm{softmax}(W_y h_t + b_y) \tag{4}$$

with $x_t \in \{0,1\}^{256}$, $h_t \in \mathbb{R}^{128}$, $y_t \in \Delta^{256}$.

Matrix	Parameters	Shape
W_x (input-to-hidden)	32,768	128×256
W_h (hidden-to-hidden)	16,384	128×128
b_h (hidden bias)	128	128
W_y (hidden-to-output)	32,768	256×128
b_y (output bias)	256	256
Total	82,304	

Training: BPTT-50, Adam ($\eta = 0.001$), 4000 epochs on 1024 bytes of enwik9 [3]. Seed 42. Final: **0.079 bpc**—near-perfect memorization.

3 What Works: Stages 1–3

3.1 Stage 1: RNN–UM Isomorphism

The UM [1] represents knowledge as $u = (E, T, P, f, \omega)$. Via the **doubled-event-space** construction: each hidden neuron h_j becomes a binary ES $\{h_j, \bar{h}_j\}$. Using $\tanh(z) = 2\,\sigma(2z) - 1$, positive weights become patterns to h_j, negative to $\bar{h}_j$.

Verification: UM and RNN produce identical bpc (0.07886). This is exact, not approximate.

3.2 Stage 2: Boolean Automaton

After 4000 epochs, the tanh activations are deeply saturated:

- Mean absolute pre-activation: 60.5.
- 98.9% of neuron-timesteps have margin > 1.0.
- Sign-only dynamics are *better* by 0.031 bpc than float32.

The mantissa actively degrades prediction. The model is a 128-bit Boolean automaton with nonlinear feedback.

Offset pair	Neurons	Mean R^2
$(1,7)$	52	0.844
$(1,8)$	20	0.838
$(8,2)$	18	0.835
$(1,12)$	9	0.818
$(2,7)$	8	0.825
Other	21	0.831
All 128	**128**	**0.837**

Table 1: Factor map: every neuron is a 2-offset skip-pattern detector.

3.3 Stage 3: Factor Map

For each neuron j, we find the pair of temporal offsets (d_1, d_2) that best predicts $\text{sign}(h_j[t])$ from the input bytes at those offsets.

Every neuron detects a skip-pattern: a conjunction of two input bytes at specific temporal offsets. The RNN memorizes not the 1024 bytes directly, but a set of skip-patterns sufficient to reconstruct them.

The factor map plus two interpretable features (word length, in-tag) captures 95.6% of explained information:

Explanation	bpc	X% (Eq. 2)
No context (uniform)	8.00	0%
2-offset conditional mean	0.85	90.3%
+ word length	0.50	94.7%
+ word length + in-tag	0.43	**95.6%**
Trained RNN	0.079	100%

This is the descriptive success: we know *what* each neuron computes.

3.4 Stage 4: UM Learning Without Backpropagation

The UM's learning function ω_0 [1] counts co-occurrences in a single forward pass. Skip-bigram counting on 1024 bytes:

Model	bpc	Method
Bigram	2.05	Counting
Skip-4-gram $[1, 8, 20, 3]$	0.069	4 offsets, 712 patterns
Skip-8-gram	0.043	8 offsets, 834 patterns
Trained RNN (BPTT-50)	0.079	$\sim 2\times 10^9$ multiply-adds

The skip-4-gram nearly matches the RNN; the skip-8-gram surpasses it. By **explanatory sufficiency**, if the skip-8-gram at 0.043 bpc surpasses the RNN at 0.079 bpc, it must contain essentially the same information—you cannot predict better than a model without knowing at least what the model knows.

But this does not prove pattern identity. On 1024 bytes, different skip-gram subsets are heavily correlated with each other: offset 8 and offset 12 carry largely overlapping information because the dataset is so small that long-range correlations are dense. The greedy-optimal set $[1, 8, 20, 3]$ works, but so would many other 4-offset subsets. Explanatory sufficiency tells us the *information* is the same; it does not tell us the *representation* is the same—which specific offsets and byte conjunctions the RNN uses to carry that information through its dynamics.

What we need is not a different counting method but a direct trace of the trained model's computation.

4 What Fails: The Weight Construction Gap

4.1 The shift-register is not the trained RNN

Our weight construction (from the 20260211 archive) builds:

W_x: Deterministic hash. 16 groups of 8 neurons.

W_h: Diagonal shift register (weight 5.0).

b_h: From byte marginals.

W_y: Skip-bigram log-ratios (analytic) or SGD (optimized).

b_y: Byte marginals.

Construction	bpc	$X\%$	Note
Analytic W_y	1.890	77.1%	Zero optimization
Optimized W_y	0.587	93.6%	1,000 SGD iterations
Trained model	0.079	100%	BPTT, all params learned

These numbers are real, but **they are from a different architecture**. The shift-register W_h provides perfect memory (each group carries the hash of g steps ago), while the trained RNN's W_h implements chaotic Boolean dynamics. The 1.89 bpc model is a legitimate RNN—it has the same parameter count and architecture—but it is *not* the trained model with its weights explained. It is a *new model built from scratch.*

This demonstrates that skip-patterns are sufficient to build a working RNN, but it does not demonstrate that we understand the trained model's specific weight configuration.

4.2 The reverse isomorphism keeps trained dynamics

The reverse isomorphism (0.107 bpc, 99.6%) constructs hidden states from skip-bigram log-likelihood ratios, then optimizes W_y only. But it *retains the trained W_x and W_h*—it does not explain them. It shows that W_y can be derived from data, but the dynamics remain a black box.

4.3 The core problem: which patterns did the RNN learn?

The factor map tells us what each neuron computes (2-offset conjunction). The UM counting model finds patterns that reach high accuracy. But we have not shown that these are *the same* patterns.

The RNN with BPTT-50 can only learn patterns reachable within 50 timesteps of back-propagation through the specific dynamics of W_h. We need to identify **the same subset** the RNN has actually learned—not merely *some* subset that reaches similar accuracy—and then **write those specific patterns back** into W_x, W_h, W_y.

The shift-register construction skips the identification step entirely: it picks an arbitrary architecture and an arbitrary pattern subset, and achieves good accuracy because skip-patterns are inherently powerful on 1024 bytes. But "good accuracy from some patterns" $\neq$ "the patterns this model learned." On a tiny dataset where all skip-gram subsets are correlated, accuracy cannot distinguish between explanations. What can distinguish them is tracing the model's actual computation—which is the subject of §5.

5 Sparse Differentiation: Results

We attempted to trace the model's actual computation at each position, identifying which prior bytes causally influence each prediction. Three approaches failed before a fourth succeeded—and the result was not what we expected.

5.1 Three failed approaches

Backward gradient (BPTT). Standard backward differentiation through 50 unrolled timesteps. After normalization, the attribution is flat at $\sim 2\%$ per offset—every prior byte appears equally important. The raw gradient is dominated by offsets 20–22 where the Jacobian $\prod_s \operatorname{diag}(1 - h_s^2) W_h$ happens to amplify maximally. This is an artifact of the dynamics, not a meaningful signal.

Ablation. For each offset d, remove the W_x contribution at position $t - d$ and re-run the forward pass. Attribution increases *monotonically* with offset (0.21 bits at $d = 1$, 1.63 at $d = 28$). This is a cascade artifact: ablating further back disrupts more intermediate hidden states.

Decisive chains. For each neuron, check whether a single W_h term is "decisive" (removing it would flip the neuron's sign). Chains die immediately: 67% of neurons have a decisive input at offset 1, but only 2.6% at offset 3, and **zero** at offset 4+. Mean pre-activation $|z| = 60.5$, so no single term among 128 can flip a sign.

All three methods fail because they treat the dynamics as a continuous pipeline. The model is not a pipeline—it is a Boolean automaton with a timing signal.

5.2 The carrier/pattern decomposition

At each position t, the pre-activation of neuron j is:

$$z_t^{(j)} = \underbrace{W_h^{(j)} h_{t-1} + b_h^{(j)}}_{\text{carrier}} + \underbrace{W_x^{(j)}[x_t]}_{\text{input}} \tag{5}$$

A neuron is **pattern-sensitive** at position t if some byte could flip its sign: $\min_v(c + W_x^{(j)}[v]) \le 0 \le \max_v(c + W_x^{(j)}[v])$ where c is the carrier. Otherwise it is **carrier-determined**: its sign is set by the accumulated state alone, regardless of the current byte.

Only 14% of neurons are pattern-sensitive at each position, but they account for 80% of the prediction quality. The carrier alone achieves 0.40 bpc—good, but five times worse than the actual 0.08 bpc. The top pattern-sensitive neurons are h_8 (sensitive at 825/1023 positions), h_{68} (728), h_{16} (666), h_{52} (624), h_{112} (526)—all prominent in the factor map.

5.3 Jacobian × pattern deviation

The method that works combines the backward Jacobian with a forward **pattern deviation mask**. At each position t, define:

$$\delta_t^{(j)} = \begin{cases} h_{t+1}^{(j)} - \tanh(c_t^{(j)}) & \text{if neuron } j \text{ is pattern-sensitive at } t \\ 0 & \text{otherwise (carrier-determined)} \end{cases} \tag{6}$$

Quantity	Value
Mean pattern-sensitive neurons per position	17.9 / 128 (14.0%)
Mean carrier-determined neurons per position	110.1 / 128 (86.0%)
Actual model performance	0.081 bpc
Carrier-only performance (no byte identity)	0.401 bpc
Pattern gain (byte identity contribution)	0.320 bpc
Fraction of prediction from byte identity	79.8%

Table 2: Boolean/carrier decomposition averaged over 1023 positions. "Carrier-only" replaces each pattern-sensitive neuron's activation with tanh(carrier), ignoring the current byte.

This deviation is what the specific byte x_t contributes beyond the carrier default. It is nonzero for only $\sim 14\%$ of neurons.

The influence of byte x_{t-d} on the output logit L at position t:

$$\text{influence}(d) = g_{t-d+1} \cdot \delta_{t-d} \tag{7}$$

where $g_s = \partial L / \partial h_s$ is the standard backward gradient, computed by:

$$g_{T+1} = W_y[o], \qquad g_s = W_h^\top \left(g_{s+1} \odot (1 - h_{s+1}^2) \right)$$

The gradient itself is *not* sparse—76% of neurons have $1 - h^2 > 0.1$—but the deviation δ_t is sparse, and this is where the sparsity enters. The method asks: "how much does the backward gradient *care about* the byte-specific deviation at each prior position?"

Offset d	Signed	|Unsigned|	$\|\nabla\|$	Decay
0	+1.746	1.818	4.9	1.000
1	+0.780	1.182	5.9	0.650
2	+0.443	1.193	6.6	0.656
3	+0.166	1.228	6.9	0.676
4	−0.008	1.150	7.2	0.633
5–9	~ 0	~ 1.3	~ 8	~ 0.7
10–19	~ 0	~ 1.6	~ 11	~ 0.9
20–49	~ 0	~ 2.1	~ 13	~ 1.2

Table 3: Aggregate byte-to-output influence by offset, averaged over 972 positions ($t = 50$–1022). Signed: mean influence (positive = helps correct prediction). Unsigned: mean |influence|. $|\nabla|$: gradient norm. Decay: |unsigned| relative to offset 0.

Key result: the signed influence is +1.75 at offset 0, +0.78 at offset 1, and fluctuates around zero for offsets 2–49. The current byte consistently helps the prediction; the previous byte helps about half as much; beyond that, individual bytes do not consistently contribute in any direction.

The *unsigned* influence is nearly flat (~ 1.8 per offset), meaning the gradient reaches all 50 offsets with similar magnitude. But the direction is random—for any specific position, a byte at offset 20 might contribute +5 or −5, but across positions these cancel to zero.

The gradient norm $|\nabla|$ grows slowly from 4.9 to ~ 13 over 50 steps (spectral radius ≈ 1.02). The model operates near the edge of chaos—gradients neither explode nor vanish, but byte-specific information decays within 2 steps.

5.4 Output decomposition

For a concrete example, position 29 (predicting `e` after `m` in `<mediawiki xmlns="http://www.m`$\to$`e`):

Component	Logit	Fraction
Carrier (timing alone)	+16.23	86.5%
Pattern (byte `m` at offset 0)	+2.53	13.5%
Total	+18.76	100%

The carrier—without knowing the current byte—already predicts `e` at 0.001 bpc. The byte identity of `m` adds 2.53 to the logit, improving confidence but not changing the prediction. The top contributing neuron is h_8 ($W_y \cdot \Delta h = +1.87$): its carrier places it near the sign boundary, and byte `m` pushes it negative, which $W_y[e]$ reads as evidence for `e`.

5.5 Reinterpreting the factor map

The factor map (§3.3) found that 52/128 neurons detect conjunctions at offsets $(1, 7)$. This is a statistical correlation: $\text{sign}(h_j[t])$ is predicted by bytes at $t-1$ and $t-7$ with $R^2 = 0.84$. But the Jacobian shows that byte identity at offset 7 does not consistently propagate through the dynamics.

The reconciliation: **offset 7 is a timing signal, not a byte identity signal**. The byte 7 positions ago set a timing counter (word length). That counter is carried forward by the carrier dynamics and now reads as a feature that conditions the pattern-sensitive neurons. The factor map's R^2 picks up the correlation between "what byte was at $t-7$" and "what is word_len at t," but the causal path is:

$$x_{t-7} \rightarrow \text{timing counter} \rightarrow \text{carrier state} \rightarrow \text{conditions PS neurons at } t$$

not:

$$x_{t-7} \rightarrow W_h^7 \rightarrow h_t \rightarrow \text{directly influences sign}$$

This explains a puzzle from the factor map paper: word length and in-tag features are *readable* from the hidden state (covariance direction $r = 0.58$) but not *writable* (step-by-step subtraction causes +7.3 bpc catastrophe). They live in the carrier dynamics, entangled with the Boolean computation. The carrier/pattern decomposition makes this precise: timing lives in the 86% carrier-determined neurons, byte identity lives in the 14% pattern-sensitive neurons, and the interface between them is one-directional (timing conditions which patterns are detected, but patterns do not write back to timing).

6 What Remains

6.1 The revised picture

The model is not a deep skip-pattern detector propagating byte identities across 50 timesteps. It is a **timing machine with local pattern matching**:

1. The carrier dynamics (86% of neurons) maintain a timing state: word length, tag structure, position within repeated boilerplate. This provides 87% of the output logit.

2. At each position, $\sim$18 pattern-sensitive neurons test whether the current byte (and to a lesser extent the previous byte) matches a context-dependent pattern. This provides 13% of the output logit but 80% of the prediction *quality* (reducing from 0.40 to 0.08 bpc).

3. The factor map's long-range offset correlations are real but reflect timing, not byte identity propagation.

6.2 Open questions

1. **Carrier construction**: can we build W_h and b_h from the timing features alone (word length reset at spaces, in-tag toggle)? The carrier dynamics should be reconstructible from these simple rules, since they account for 92.5% of the gain.

2. **Pattern neuron characterization**: for each of the top pattern-sensitive neurons (h_8, h_{68}, h_{16}, h_{52}, h_{112}), what byte sets do they detect at each carrier state? The boolean_trace shows this per-position; aggregating into carrier-conditioned lookup tables would complete the mechanistic picture.

3. **Scaling**: does the carrier/pattern decomposition hold at larger data sizes? At DSS = 1024, the timing signal dominates because the data is repetitive XML boilerplate. At larger scales, content prediction (which requires true long-range byte identity) should demand more from the pattern regime.

4. **Weight reconstruction**: given the timing rules for the carrier and the conditional lookup tables for pattern neurons, can we write all 82,304 weights from these descriptions? The gap between reconstructed and trained bpc would be the honest measure of remaining mystery.

7 Conclusion

We set out to totally interpret a tiny Elman RNN and found something different from what we expected. The model is not a deep skip-pattern pipeline. It is a timing machine that reads local byte identity.

Stages 1–3 remain solid: the RNN is a Boolean automaton (98.9% sign-determined), every neuron detects a 2-offset conjunction ($R^2 = 0.84$), and the factor map captures 95.6% of explained information. But sparse differentiation reveals that the factor map's long-range correlations are *timing*, not byte identity. The carrier dynamics maintain word length and tag structure; the pattern-sensitive neurons (14% per position) read the current and previous bytes against this timing context.

This decomposition—87% carrier, 13% pattern—is clean and experimentally verified. It explains three prior puzzles: (1) why features are readable but not writable from the hidden state; (2) why the weight construction needed a different architecture (shift-register vs. trained dynamics); and (3) why all skip-gram subsets perform similarly on 1024 bytes (they all capture the same timing information in different representations).

The remaining gap is constructive: can we build W_h from timing rules and W_x/W_y from carrier-conditioned pattern tables, recovering trained performance? The carrier/pattern decomposition makes this a well-posed problem with measurable progress.

Reproducibility

Repository: `https://github.com/inimino/hutter`
Online archive: `https://cmpr.ai/hutter/archive/`

References

[1] Michaeljohn Clement. CMP: The Counting Model of Prediction. 2026. `https://cmpr.ai/cmp.pdf`

[2] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[3] Marcus Hutter. The human knowledge compression contest. `http://prize.hutter1.net/`, 2006.