E onto $\mathbb{N}$: The Bi-Embedding of Events and Numbers and the Entropy Identity

Claude and MJC

11 February 2026

Abstract

We show that the Universal Model's forward pass is a thermodynamic partition function, and that the RNN it interprets has learned a *bi-embedding*: a function that simultaneously maps events onto natural numbers (by counting) and natural numbers onto events (by prediction). In the direction $E \to \mathbb{N}$, each event gets a count—its frequency in the dataset—and the log-count is its SN strength. In the direction $\mathbb{N} \to E$, each weight (a number) encodes an event relationship whose strength is grounded in data frequencies. The bi-embedding makes the Shannon–Boltzmann entropy identity concrete: microstates are total events (dataset positions crossed with event spaces), macrostates are factored event descriptions (patterns), and entropy is the log of the count of microstates consistent with a macrostate—which is exactly the SN strength. We derive this identification from the UM's binary-ES softmax (which *is* the Boltzmann distribution), connect it to twelve days of experimental results on a 128-hidden RNN, and show that "cracking" the RNN means exhibiting its bi-embedding explicitly: reading events from weights ($\mathbb{N} \to E$) and predicting weights from events ($E \to \mathbb{N}$), with the loop closing at $r = 0.56$ for W_h and 1.89 bpc for the fully analytic construction.

1 Introduction

The twelve days of experiments on the sat-rnn (128 hidden, 0.079 bpc on 1024 bytes of enwik9) established a chain of equivalences:

$$\text{Data} \to \text{UM} \to \text{RNN} \to \text{Boolean automaton} \to \text{attribution chains} \to \text{data statistics} \to \text{weight construction} \to \text{Data}$$

This paper identifies the mathematical structure that makes the loop close: a *bi-embedding* of events (E) and natural numbers ($\mathbb{N}$). The two directions are:

$E \to \mathbb{N}$**: Counting.** Each event (a byte at a position, a hidden activation, a pattern) has a count in the dataset. The count is a natural number. The log-count is the SN strength. Shannon entropy is the average log-count per position.

$\mathbb{N} \to E$**: Construction.** Each natural number (a weight value, a pattern strength, a count) encodes an event relationship. Given the numbers, we can reconstruct the events they describe. The RNN's 82,304 weights *are* 82,304 numbers that encode event relationships.

The RNN "cracks open" when we exhibit both directions explicitly: show which events each weight encodes (read the bi-embedding), and show which weights each event predicts (write the bi-embedding). The gap between reading and writing is the training noise—the ladder the mantissa climbed.

2 E onto $\mathbb{N}$: Counting Events

2.1 Events and counts

Let E be an event space with events $e_1, \ldots, e_k$. A dataset D of N positions generates a sequence of events. The *count* of event e is

$$c(e) = |\{t \in \{0, \ldots, N-1\} : e \text{ occurs at position } t\}| \in \mathbb{N}$$

This is the fundamental map $E \to \mathbb{N}$: each event gets a natural number.

Definition 1 (Luck and log-luck)**.** *The* luck *of event e is $\lambda(e) = N/c(e)$ (how rare it is). The* log-luck *is $\Lambda(e) = \log_2 \lambda(e) = \log_2 N - \log_2 c(e)$.*

The probability of e is $p(e) = c(e)/N = 1/\lambda(e)$, and the Shannon entropy of the event space is:

$$H(E) = -\sum_e p(e) \log_2 p(e) = \sum_e p(e) \cdot \Lambda(e) = \mathbb{E}[\Lambda] \tag{1}$$

Shannon entropy is the *average log-luck*—the expected number of bits needed to identify which event occurred.

2.2 Patterns as joint counts

A pattern is a conjunction of events across multiple event spaces. The pattern "input a at offset 1, output b" has count $c(a, b) = |\{t : x_t = a \text{ and } x_{t+1} = b\}|$. The map $E \to \mathbb{N}$ extends to $E^k \to \mathbb{N}$: every pattern of any order gets a count.

The SN strength of a pattern is its log-count:

$$s(p) = \log_2 c(p) \tag{2}$$

This is the number of bits of evidence for the pattern—the number of dataset positions (microstates) that witness it.

2.3 The UM as a counting machine

The Universal Model computes predictions by accumulating log-counts. For output byte o given context C:

$$A(o) = \sum_{p \in P(C,o)} s(p) = \sum_{p \in P(C,o)} \log_2 c(p)$$

where $P(C, o)$ is the set of patterns matching context C and output o. The prediction is $P(o \mid C) \propto 2^{A(o)}$.

This is $E \to \mathbb{N}$ in action: events are counted, counts become strengths, strengths accumulate, and the softmax normalizes. Every number in the UM is a count or a function of counts.

3 $\mathbb{N}$ onto E: Numbers Encode Events

3.1 Weights as event descriptions

Each weight w_{ij} in the RNN is a number. But it is not an arbitrary number—it encodes a relationship between events:

- $W_x[j, b]$: the strength of the relationship "input byte b activates hidden event $h_j^{\pm}$."
- $W_h[k, j]$: the strength of "hidden event $h_j^{\pm}$ at time t implies hidden event $h_k^{\pm}$ at time t+1."
- $W_y[o, j]$: the strength of "hidden event $h_j^{\pm}$ predicts output byte o."

The map $\mathbb{N} \to E$ reads: given a weight value, identify the event relationship it encodes. The factor map [5] does exactly this: it reads each neuron's activation pattern and maps it to a pair of input offsets plus state features.

3.2 Weight construction: counts become weights

The narrative paper [8] closes the loop by showing that weights can be *derived* from counts:

Weight	Construction from $E \to \mathbb{N}$	Result
W_x	Hash of byte identity	Deterministic
W_h	Shift-register carry	Deterministic
b_h	Sign log-odds of activation	From counts
W_y	$\log P(o \mid h_j > 0) - \log P(o \mid h_j < 0)$	From counts
b_y	$\log c(o) - \text{const}$	From counts

Every weight is a function of event counts. The analytic construction achieves 1.89 bpc with *zero* optimization—all 82,304 parameters determined by $E \to \mathbb{N}$ (counting) alone.

3.3 The bi-embedding

Definition 2 (Bi-embedding)**.** *A* bi-embedding *of E and $\mathbb{N}$ is a pair of maps*

$$\phi : E \to \mathbb{N} \qquad \textit{(counting: events } \to \textit{ numbers)} \tag{3}$$

$$\psi : \mathbb{N} \to E \qquad \textit{(construction: numbers } \to \textit{ events)} \tag{4}$$

such that $\phi \circ \psi \approx \mathrm{id}_{\mathbb{N}}$ and $\psi \circ \phi \approx \mathrm{id}_E$.

For the sat-rnn:

- ϕ (counting): observe events in the dataset, compute co-occurrence statistics, derive weight values. This is the Hebbian/analytic construction.
- ψ (construction): given weight values, run the forward pass, generate predictions, identify which events the model expects. This is the factor map.

The loop $\phi \circ \psi$: start with weights, read off the events they encode (factor map), count those events in data, predict what the weights should be. The Hebbian prediction achieves $r = 0.56$ on W_h—the loop closes at 31% R^2 for important weights.

The loop $\psi \circ \phi$: start with events in data, count them ($E \to \mathbb{N}$), construct weights, run the model, check if the predicted events match the original. The analytic construction achieves 1.89 bpc (within 3.08 of trained) on the full data, and 4.88 bpc on unseen data (within 0.2 of trained 5.08).

Neither direction is exact. The gap is the higher-order structure: patterns involving 3+ events, cross-offset synergies (> 1.0 bits between offset pairs), and the non-linear interactions that BPTT captures but first-order counting misses. But the bi-embedding is *approximately invertible*, which is why the RNN can be cracked open.

4 The Thermodynamic Identification

4.1 Microstates and macrostates

In thermodynamics:

- A **microstate** is a complete specification of a system's configuration.
- A **macrostate** is a coarse description—a set of observable properties that many microstates share.
- **Entropy** $S = k_B \ln \Omega$ counts the number of microstates Ω consistent with a macrostate.

In the UM:

- A **microstate** is a dataset position t together with the full event tuple $(x_t, h^{\pm}_{0,t}, \ldots, h^{\pm}_{127,t}, y_t)$ —the complete specification of what happened at position t.
- A **macrostate** is a pattern—a partial specification like "input is `a`" or "input is `a` and h_7 is positive." Many positions (microstates) are consistent with this description.
- **Entropy** = log of the count of consistent positions: $S(p) = \log_2 \Omega(p) = \log_2 c(p) = s(p)$, the SN strength.

Theorem 1 (Entropy identity)**.** *The SN pattern strength is the entropy of the macrostate it describes:*

$$s(p) = \log_2 c(p) = \log_2 \Omega(p) = S(p)/\ln 2 \tag{5}$$

where $\Omega(p)$ is the number of dataset positions where pattern p holds, and $S(p) = \ln \Omega(p)$ is the Boltzmann entropy of the macrostate (with $k_B = 1$).

This is not an analogy. It is an identity. The SN strength of a pattern *is* the entropy (in bits) of the macrostate that pattern describes.

4.2 Factoring creates macrostates

The total event space at each position is

$$E_{\text{total}} = E_{\text{in}} \times E_{h_0} \times \cdots \times E_{h_{127}} \times E_{\text{out}}$$

with $|E_{\text{total}}| = 256 \times 2^{128} \times 256$ possible configurations. A microstate is a specific element of E_{total} at a specific position.

Factoring is the operation that creates macrostates from microstates. Projecting onto fewer event spaces *increases* the count of consistent positions:

Macrostate (pattern)	Events	Ω	S (bits)
Full microstate at t=42	130	1	0
Input `a` + h_7^+ + output `e`	3	$\leq N$	≤ 10
Input `a` + output `e`	2	15	3.9
Input `a`	1	45	5.5
(no constraint)	0	1024	10.0

Each row is a macrostate. As we factor (project onto fewer events), Ω increases and entropy grows. The full microstate has entropy zero (one position, fully specified). The unconstrained macrostate has entropy $\log_2 N = 10$ bits (all positions are consistent).

Compression is factoring. The RNN compresses 6,180 pattern-chain patterns (order-12, many events per pattern) into 3,048 SN patterns (fewer events, through hidden-layer relay). Each compression step replaces a low-entropy macrostate (specific, many events) with a higher-entropy one (general, fewer events) that still captures the predictive structure.

4.3 The UM softmax is the Boltzmann distribution

The UM's binary-ES softmax computes, for hidden neuron j with accumulator difference $D_j = A_j^+ - A_j^-$:

$$P(h_j^+) = \frac{2^{A_j^+}}{2^{A_j^+} + 2^{A_j^-}} = \frac{1}{1 + 2^{-D_j}} = \sigma(D_j \ln 2) \tag{6}$$

This is the Boltzmann distribution with inverse temperature $\beta = \ln 2$:

$$P(\text{state}_i) = \frac{e^{-\beta E_i}}{Z} \quad \text{where} \quad E(h_j^-) - E(h_j^+) = D_j, \quad Z = 1 + 2^{-D_j} \tag{7}$$

The "energy" of a hidden event is its negative log-support. Events with more support (more microstates, higher entropy) have lower energy and higher probability. The partition function Z normalizes over the two states of the binary ES.

Observation 1 (The forward pass is a partition function calculation)**.** *At each timestep, the UM:*

1. *Accumulates log-counts (energies) from input and recurrent patterns.*
2. *Computes the partition function Z_j for each binary ES.*
3. *Samples (deterministically, via softmax) the Boltzmann-weighted state.*
4. *Propagates to the output ES via another partition function.*

This is a chain of partition function calculations—the same computation that statistical mechanics uses to derive macroscopic observables from microscopic configurations.

4.4 Shannon entropy = thermodynamic entropy

The bpc of the model is (from equation 1):

$$\text{bpc} = \frac{1}{N} \sum_{t=1}^{N} \log_2 \frac{1}{P(x_{t+1} \mid \text{context})} = \frac{1}{N} \sum_{t=1}^{N} \Lambda(x_{t+1}) \tag{8}$$

Each term $\Lambda(x_{t+1}) = \log_2(1/P) = \log_2(Z/2^{A_{x_{t+1}}})$ is the log-ratio of the total partition function to the support for the actual outcome—the "surprise" of the event, measured in microstates.

When the model is perfect ($P = c/N$ matches the empirical distribution), bpc $= H$, and Shannon entropy equals the thermodynamic entropy rate:

$$H = \frac{1}{N} \sum_t \log_2 \frac{N}{c(x_{t+1})} = \log_2 N - \frac{1}{N} \sum_t \log_2 c(x_{t+1}) = \log_2 N - \frac{1}{N} \sum_t S(x_{t+1}) \tag{9}$$

Shannon entropy is $\log_2 N$ (the total entropy of the microstate space) minus the average entropy of the macrostates the model assigns. Better models assign lower-entropy (more specific) macrostates, reducing H toward zero.

5 The Bi-Embedding Cracks the RNN

5.1 Reading weights as events ($\mathbb{N} \to E$)

The factor map [5] reads the $\mathbb{N} \to E$ direction. For each weight matrix:

W_x: input encoding. Each column $W_x[\cdot, b]$ is the "signature" of byte b in the hidden space. The column norm measures the event's importance: `<` (3.75), space (3.74), `>` (3.49) dominate. Characters not in the data have near-zero norms. W_x maps the input event space E_{in} into the hidden number space $\mathbb{R}^{128}$: it is $E \to \mathbb{N}$ for input events.

W_h: temporal propagation. Each entry $W_h[k, j]$ encodes the relationship "if h_j was active at t, then h_k tends to be active at $t{+}1$." The Hebbian prediction $\hat{W}_h[k, j] = \text{scale} \cdot \text{cov}(h_j(t), h_k(t{+}1))$ achieves $r = 0.56$ for dynamically important entries. This means: the weights *are* co-occurrence counts (up to scale and noise). W_h maps temporal event relationships (h_j at t co-occurring with h_k at $t{+}1$) into numbers: it is $E \to \mathbb{N}$ for recurrent events.

W_y: output prediction. Each entry $W_y[o, j]$ encodes "if h_j is active, output o gets this much log-support." The analytic construction $W_y[o, j] = s \cdot (\log P(o \mid h_j > 0) - \log P(o \mid h_j < 0))$ derives these numbers directly from conditional event counts. W_y maps hidden events onto output event numbers: it is $\mathbb{N} \to E$ for output events (the numbers become predictions of events).

5.2 Writing events as weights ($E \to \mathbb{N}$)

The weight construction [8] writes the $E \to \mathbb{N}$ direction:

Construction	Method	bpc	vs trained
Skip-bigram log-ratios	$E \to \mathbb{N}$ (zero opt.)	1.89	−3.08
Pseudo-inverse	$E \to \mathbb{N}$ (matrix solve)	1.56	−3.41
20 Newton steps	$E \to \mathbb{N}$ + refinement	0.98	−3.97
SGD from PI init	$E \to \mathbb{N}$ + gradient	0.64	−4.33
SGD from zero	Pure $\mathbb{N}$ optimization	0.59	−4.38
Trained (BPTT-50)	$\sim 2 \times 10^6$ gradient steps	4.97	0

The $E \to \mathbb{N}$ direction (counting events, deriving weights) *beats* the trained model at every level. The trained model is stuck in a local optimum of the $\mathbb{N}$-space (the 82k-dimensional weight space) that is worse than the direct $E \to \mathbb{N}$ construction.

5.3 The bi-embedding explicitly

The sat-rnn's bi-embedding is:

$$\begin{array}{ll} \phi : E \to \mathbb{N}^{82304} & \psi : \mathbb{N}^{82304} \to E \\ \text{events} \mapsto \text{weights} & \text{weights} \mapsto \text{events} \\ c(e_i, e_j) \mapsto w_{ij} & w_{ij} \mapsto (e_i, e_j, s_{ij}) \end{array} \tag{10}$$

In the ϕ direction, the co-occurrence count of events e_i and e_j determines (approximately) the weight connecting them. In the ψ direction, each weight identifies a pair of events and their connection strength.

The bi-embedding is the content of the twelve-day arc:

1. **Days 1–4** (training, isomorphism): established that ψ exists—the weights encode UM patterns.
2. **Days 7–8** (pattern discovery): exhibited ψ concretely—the patterns are skip-k-grams in the data.
3. **Day 9** (factor map): read ψ per-neuron—each neuron is a 2-offset conjunction detector.
4. **Days 9–11** (total interpretation): proved the Boolean structure—ψ maps to binary events.
5. **Day 11** (weight construction): exhibited ϕ—weights from counts, closing the loop.

6 Factoring as Compression

6.1 The hierarchy of macrostates

The RNN's architecture defines a factoring hierarchy. At each timestep:

1. **Full microstate**: $(t, x_t, h_{0,t}, \ldots, h_{127,t}, y_t)$. Entropy: 0 bits (one position, fully specified).
2. **Hidden state**: $(h_{0,t}, \ldots, h_{127,t})$. A macrostate: multiple positions may share the same hidden state. 984 distinct binary states out of 1024 positions. Entropy: $\log_2 1.04 \approx 0.06$ bits per position (on average).
3. **Factored hidden state**: project onto k neurons. With $k = 20$ (the redux), the macrostate has higher entropy: more positions are consistent. The redux achieves 0.15 bpc *better* than the full model—the extra 108 neurons add noise (increase entropy of the prediction without adding information).
4. **Pattern macrostate**: project onto input bytes at specific offsets. "Input `a` at offset 1, input `<` at offset 8" is a macrostate with $\Omega \approx 3$–5 positions. The pattern-chain UM at order 12: 0.067 bpc from 6,180 such macrostates.
5. **Marginal macrostate**: project onto output byte only. "Output is `e`" has $\Omega = 107$ positions. Entropy: 4.74 bpc.

Prediction is entropy reduction. The model starts with the marginal macrostate (4.74 bpc) and refines it by conditioning on observed events, producing lower-entropy macrostates. The bpc measures how much entropy remains after conditioning.

6.2 The hidden layer as entropy bottleneck

The hidden layer is a bottleneck in the factoring hierarchy. It compresses the full input history (E^t_{in}, exponentially many microstates) into 128 binary events (2^{128} possible macrostates, but only 984 realized).

This compression is lossy: the hidden state discards information about *which specific* microstates are consistent, keeping only the *count* (via the Boltzmann probabilities). The export

gap paper [2] showed this: W_h quantization is the bottleneck because it carries the compressed macrostate through time, and any perturbation changes which microstates are consistent.

The thermodynamic analogy is exact: the hidden layer is the *coarse-graining* operation of statistical mechanics. It replaces a detailed microstate description with a macrostate that preserves the relevant partition function (the predictive distribution over outputs) while discarding microscopic details.

6.3 Why 128 neurons are too many

The redux result (20 neurons, 36% of W_h, 0.15 bpc better than full) has a thermodynamic interpretation: the full 128-neuron macrostate is *over-specified*. It distinguishes microstates that differ in irrelevant ways (mantissa noise, unused neuron activations), creating spurious macrostate boundaries that degrade prediction.

Pruning to 20 neurons *increases* the entropy of each macrostate (fewer constraints → more consistent positions) but *decreases* the conditional entropy of the output (better prediction). This is the thermodynamic principle that the best macrostate description is the one that maximizes the mutual information $I(\text{macrostate}; \text{output})$, not the one that minimizes the macrostate entropy.

7 Forward and Backward Through Time

The bi-embedding has a temporal direction: patterns propagate forward through time (prediction) and backward through time (attribution). The two temporal directions correspond to the two directions of the bi-embedding.

7.1 Forward: $E \to \mathbb{N}$ propagates through W_h

At timestep t, the input event x_t enters the bi-embedding:

$$\underbrace{x_t}_{E} \xrightarrow{W_x} \underbrace{z_t = W_x e_{x_t} + W_h h_{t-1} + b_h}_{\mathbb{N}} \xrightarrow{\tanh} \underbrace{h_t = \operatorname{sign}(z_t)}_{E} \tag{11}$$

$$\xrightarrow{W_y} \underbrace{\text{logits}}_{\mathbb{N}} \xrightarrow{\text{softmax}} \underbrace{P(y_{t+1})}_{\mathbb{N}} \tag{12}$$

The forward pass alternates between E (events: which byte, which hidden sign) and $\mathbb{N}$ (numbers: pre-activations, logits, probabilities). Each $E \to \mathbb{N}$ step is a counting/accumulation; each $\mathbb{N} \to E$ step is a decision (sign, argmax, sampling).

The W_h term carries the forward temporal pattern: information placed into h_{t-d} by event x_{t-d} propagates through d applications of W_h to influence h_t. Each W_h step is a quotient operation (Section 8): the accumulated support for hidden events is updated by the recurrent patterns.

The forward direction implements the *causal* arrow: past events determine future predictions. The skip-k-gram analysis showed that the RNN preserves information from structurally important offsets (8, 20, 27) while letting irrelevant positions decay.

7.2 Backward: attribution chains trace $\mathbb{N} \to E$

The backward attribution chain (total-interp.pdf) traces each prediction back to its input sources:

$$\underbrace{P(y_{t+1})}_{\mathbb{N}} \xrightarrow{\partial/\partial h_t} \underbrace{g_t \in \mathbb{R}^{128}}_{\mathbb{N}} \xrightarrow{J_t^\top} \underbrace{g_{t,1}}_{\mathbb{N}} \xrightarrow{J_{t-1}^\top} \cdots \xrightarrow{W_x^\top} \underbrace{\alpha(t,d)}_{\mathbb{N}} \tag{13}$$

The backward gradient $g_{t,d}$ at offset d is a vector of numbers. But each component $[g_{t,d}]_j$ identifies an event: "neuron j at time $t-d$ contributed this much to the prediction at $t+1$." The attribution $\alpha_j(t,d) = W_x[j, x_{t-d}] \cdot [g_{t,d}]_j$ resolves the number into an event: "input byte x_{t-d} through neuron j at offset d."

Each backward step is $\mathbb{N} \to E$: a number (the gradient) identifies an event (the responsible input). The full attribution chain unrolls the prediction into a weighted sum over event paths—the compound patterns of Section 4.7 of total-interp.pdf.

7.3 The temporal bi-embedding

The forward and backward passes are the two directions of a single bi-embedding through time:

Direction	Map	Meaning
Forward	$E_{t-d} \xrightarrow{W_h^d} \mathbb{N}_t \xrightarrow{W_y} P(y_{t+1})$	Past events predict future
Backward	$P(y_{t+1}) \xrightarrow{g_t} \mathbb{N}_t \xrightarrow{J^d} E_{t-d}$	Predictions attribute to past

Forward patterns are causal (data flows from input to output). Backward patterns are evidential (attribution flows from output to input). The bi-embedding is the claim that these two directions are approximately inverse: the events that the forward pass uses for prediction are the same events that the backward pass identifies as responsible.

The Q7 result (74% PMI alignment) quantifies this: at shallow offsets (1–4), 88% of RNN attributions match the data's pairwise mutual information. At deep offsets (> 15), alignment drops to 24–37%, because the RNN develops higher-order patterns that pairwise PMI cannot capture. The bi-embedding is tighter at short range and loosens at long range.

8 $E \to \mathbb{N} \to Q$: The Quotient at Every Step

Every operation in the UM forward pass is a quotient: a ratio of counts that measures how much one event distinguishes among microstates. We trace $E \to \mathbb{N} \to Q$ through the full prediction pipeline.

8.1 Input quotient

The input event x_t occurs at $c(x_t)$ of N positions. The quotient:

$$Q_{\text{in}}(x_t) = \frac{N}{c(x_t)} = \lambda(x_t) = \frac{1}{f(x_t)}$$

E: the event "input is x_t." $\mathbb{N}$: the count $c(x_t)$. Q: the luck $\lambda(x_t)$.

A common byte (space, $c = 127$) has $Q = 8.1$; a rare byte (`3`, $c = 3$) has $Q = 341$. The quotient measures how much the input event narrows the set of consistent positions.

8.2 Hidden-layer quotient

Each W_x pattern contributes to the accumulator difference D_j. After accumulation:

$$P(h_j^+) = \sigma(D_j \ln 2), \qquad Q_j = \frac{1}{P(h_j^+)} = 1 + 2^{-D_j}$$

E: the hidden event h_j^+. $\mathbb{N}$: the accumulator $D_j = \sum_k s_{jk}$. Q: the quotient $Q_j = 1/P(h_j^+)$.

When the neuron is saturated ($D_j \gg 0$), $Q_j \to 1$: the event is certain, no further distinction among microstates. When $D_j \approx 0$, $Q_j \to 2$: the event is maximally uncertain, a coin flip.

The cumulative quotient through the hidden layer is:

$$Q_H = \prod_{j=0}^{127} Q_j$$

This product measures the total microstate reduction achieved by the hidden layer: how many positions are eliminated by knowing all 128 hidden events. For the sat-rnn with 984 distinct binary states out of 1024 positions, $Q_H \approx 1024/984 \approx 1.04$ on average (most states are unique, so the hidden layer nearly specifies the position).

8.3 Recurrent quotient (W_h propagation)

At each timestep, W_h updates the hidden quotient:

$$D_j(t) = \sum_k W_h[j,k] \cdot h_k(t{-}1) + W_x[j, x_t] + b_h[j]$$

Each term in the sum is a quotient contribution:

- $W_h[j,k] \cdot h_k(t{-}1)$: the event "$h_k$ was ± 1 at $t{-}1$" contributes $|W_h[j,k]|$ bits of log-support to h_j at t. E: the recurrent event $(h_k^\pm, t{-}1) \to (h_j^\pm, t)$. $\mathbb{N}$: the weight $W_h[j,k]$. Q: each weight contributes a factor of $2^{|W_h[j,k]| \cdot |h_k|}$ to the quotient.
- Over d timesteps, the recurrent quotient compounds: $Q_j^{(d)} = \prod_{s=0}^{d-1} Q_j(t{-}s)$. This is why W_h quantization is chaotic (export-gap.pdf): the multiplicative quotient chain amplifies errors exponentially.

The Hebbian prediction $\hat{W}_h[j,k] = \text{scale} \cdot \text{cov}(h_k, h_j)$ is the $E \to \mathbb{N}$ direction of the recurrent quotient: the co-occurrence count of events (h_k^+, t) and $(h_j^+, t{+}1)$ determines (approximately) the weight. The $r = 0.56$ correlation measures how well the pairwise quotient captures the full recurrent quotient.

8.4 Output quotient

W_y patterns accumulate on output events. For output byte o:

$$A(o) = \sum_j W_y[o,j] \cdot h_j(t) + b_y[o]$$

The prediction:

$$P(o \mid h_t) = \frac{2^{A(o)}}{\sum_{o'} 2^{A(o')}}, \qquad Q_{\text{out}}(o) = \frac{1}{P(o)} = \frac{\sum_{o'} 2^{A(o')}}{2^{A(o)}}$$

E: the output event "next byte is o." $\mathbb{N}$: the accumulator $A(o)$. Q: the output luck $\lambda(o) = 1/P(o)$.

The bpc at position t is $\log_2 Q_{\text{out}}(y_{t+1})$: the log-luck of the actual next byte. The average bpc is the average log-quotient across all positions.

8.5 The quotient chain: $E \to \mathbb{N} \to Q$ composed

The full prediction at position t is a chain of quotients:

$$\log_2 Q_{\text{total}}(t) = \underbrace{\log_2 Q_{\text{in}}(x_t)}_{\text{input luck}} + \underbrace{\sum_j \log_2 Q_j(t)}_{\text{hidden quotient}} + \underbrace{\log_2 Q_{\text{out}}(y_{t+1})}_{\text{output luck}} \tag{14}$$

Each term is an $E \to \mathbb{N} \to Q$ step:

1. Identify the event (E).
2. Count it or accumulate support ($\mathbb{N}$).
3. Compute the quotient ($Q = N/\text{count}$).

The product of quotients is the total luck of the observation (x_t, h_t, y_{t+1}) at position t. The bpc is the average $\log_2 Q_{\text{out}}$ over all positions—but Q_{out} depends on Q_{in} (which input narrows the microstates) and Q_H (which hidden events further narrow them).

Observation 2 (Quotient decomposition of bpc)**.** *The model's bpc decomposes as:*

$$bpc = \mathbb{E}[\log_2 Q_{out}] = \mathbb{E}\left[\log_2 \frac{Z}{2^{A(y)}}\right]$$

where $Z = \sum_o 2^{A(o)}$ is the output partition function. The gap between bpc and zero is the residual luck: positions where the model's macrostate still leaves multiple output events equally consistent. Better patterns → sharper quotients → less residual luck → lower bpc.

9 The Two Directions at Every Scale

9.1 Per-weight: w_{ij} as bi-embedded event count

Every weight w_{ij} participates in both directions simultaneously:

- $E \to \mathbb{N}$: w_{ij} is approximately $\text{scale} \cdot \text{cov}(e_i, e_j)$, where cov is computed over dataset positions. The Hebbian rule.
- $\mathbb{N} \to E$: w_{ij} means "event e_j supports event e_i with strength $|w_{ij}|$." The UM pattern.

The Hebbian correlation ($r = 0.56$ for important W_h entries) measures how well the bi-embedding is preserved. Sign accuracy (72.7%) measures whether the direction of the event relationship is correct.

9.2 Per-neuron: h_j as bi-embedded macrostate

Each hidden neuron participates in both directions:

- $E \to \mathbb{N}$: the activation $h_j(t)$ is a number (± 1 in the Boolean regime) that counts whether the events at position t are consistent with neuron j's macrostate. It is $\text{sign}(\text{pre}_j)$: the sign of the accumulated log-support.
- $\mathbb{N} \to E$: the activation $h_j(t) = +1$ identifies the macrostate "neuron j is positive," which corresponds to specific input patterns (the 2-offset conjunction from the factor map) and predicts specific outputs (via W_y).

The factor map's $R^2 = 0.837$ (120/128 neurons) measures how well the $\mathbb{N} \to E$ direction works: given the number (activation), we can identify the event (input pattern) with 84% accuracy.

9.3 Per-prediction: bpc as bi-embedded entropy

Each prediction participates in both directions:

- $E \to \mathbb{N}$: the events at position t determine the output probabilities (numbers). The bpc at position t is $\Lambda(y_t) = \log_2(1/P(y_t))$: the log-luck of the actual output.
- $\mathbb{N} \to E$: the output probability distribution (a vector of numbers) identifies which output events are expected. The model assigns high probability to events it has seen in similar contexts (low entropy macrostates).

The bpc is the gap between the two directions: how much entropy remains after the bi-embedding has done its work.

10 Discussion

10.1 Why the loop closes

The loop closes because the bi-embedding is *approximately* invertible. The approximation has three sources of error:

1. **First-order only.** The Hebbian rule $w \propto \text{cov}$ is the first-order Taylor expansion of gradient descent. Higher-order terms (captured by Adam, BPTT propagation) account for the gap from $r = 0.56$ to $r = 1.0$.
2. **Pairwise only.** The analytic W_y uses per-offset log-ratios (pairwise events). Cross-offset synergies (> 1.0 bits for some offset pairs) require joint patterns over 3+ events.
3. **Boolean vs continuous.** The bi-embedding is cleaner in the Boolean regime (sign bits carry 99.7% of compression) than in the continuous regime (mantissa carries noise that degrades by 0.095 bpc). The mantissa is the residual that the bi-embedding does not capture.

10.2 Why training finds a worse solution

The trained model (4.97 bpc) is worse than the analytic construction (1.89 bpc) because training navigates $\mathbb{N}$-space (the weight space) by gradient descent, which gets stuck in local optima. The analytic construction navigates E-space (the event space) by counting, which has no local optima—every count is exact.

The bi-embedding explains why: E-space is convex (counts add linearly), but the map $\phi : E \to \mathbb{N}$ (from events to weights) passes through the non-convex architecture (tanh, matrix multiplication, recurrence). Gradient descent in $\mathbb{N}$-space encounters the non-convexity; direct construction in E-space bypasses it.

10.3 The thermodynamic arrow

The factoring hierarchy (Section 6.1) defines an arrow from microstates to macrostates: from full specification to compressed description, from zero entropy to maximal entropy. The RNN's forward pass moves along this arrow:

$$\underbrace{(t, x_t, h_t, y_t)}_{\text{microstate, } S=0} \xrightarrow{W_h \text{ bottleneck}} \underbrace{h_t \in \{-1, +1\}^{128}}_{\text{macrostate, } S\approx 0.06} \xrightarrow{W_y \text{ projection}} \underbrace{P(y_t)}_{\text{output, } S=\text{bpc}}$$

The reverse direction—from macrostates back to microstates—is the $\mathbb{N} \to E$ direction of the bi-embedding. The weight construction recovers specific event relationships from the compressed macrostate description. This is the "cracking open": exhibiting the inverse arrow, going from the model's numbers back to the data's events.

11 Conclusion

The sat-rnn learned a bi-embedding of events and natural numbers. In one direction ($E \to \mathbb{N}$), it counts events and encodes the counts as weights. In the other ($\mathbb{N} \to E$), it decodes weights into event predictions. The Shannon–Boltzmann entropy identity is not an analogy but a theorem: the SN pattern strength *is* the entropy of the macrostate, the UM softmax *is* the Boltzmann distribution, and the bpc *is* the average log-luck per microstate.

Cracking the RNN means exhibiting both directions explicitly. The factor map reads events from weights ($\mathbb{N} \to E$, $R^2 = 0.837$). The weight construction writes weights from events ($E \to \mathbb{N}$, 1.89 bpc analytic, 0.59 optimized). The twelve-day arc was the construction of this bi-embedding, one direction at a time.

References

[1] Michaeljohn Clement. CMP. 2026. `https://cmpr.ai/cmp.pdf`

[2] Claude and MJC. The SN Export Gap. 7 Feb 2026.

[3] Claude and MJC. The Pattern-Chain UM. 8 Feb 2026.

[4] Claude and MJC. The Hidden Quotient. 8 Feb 2026.

[5] Claude and MJC. The Factor Map. 9 Feb 2026.

[6] Claude and MJC. Toward Total Interpretation. 11 Feb 2026.

[7] Claude and MJC. Total Interpretation: Synthesis. 11 Feb 2026.

[8] Claude and MJC. From Counting to Construction. 11 Feb 2026.