The SN Export Gap: Byte-Weight Mapping for Binary ES Softmax

Claude and MJC

7 February 2026

Abstract

We investigate mapping RNN weights to byte-valued SN pattern strengths $[0, 255]$ so that the UM forward pass (additive patterns with binary-ES softmax between layers) reproduces the RNN's inference. We derive the matching condition ($D_j = 2\,\mathrm{pre}_j / \ln 2$), identify W_h as the bottleneck via layer isolation, and find that 8-bit resolution is adequate per-weight but recurrent error amplification makes results chaotic. This chaotic sensitivity is an artifact: the RNN was trained with BPTT-50, so error accumulation beyond 50 steps is uncontrolled. We give a Bayesian derivation via $Q = \lambda$, treating dataset positions as microstates, and introduce the structural constraint that all UM patterns must chain from input bytes to output bytes—establishing the data-term framework for subsequent interpretation.

1 Binary ES Softmax

1.1 The Matching Condition

Each hidden neuron j has a binary ES $\{h_j^+, h_j^-\}$ with accumulators A_j^+ and A_j^-. Patterns contribute additively; then softmax determines the activation. With base-2 log-support (strengths are binary log values):

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

The effective activation is $h_j = 2P(h_j^+) - 1$.

From the doubled-E isomorphism (20260131), the RNN's activation is $h_j = \tanh(\mathrm{pre}_j)$ where $\tanh(x) = 2\sigma(2x) - 1$. Setting equal:

$$\sigma\big(D_j \cdot \ln 2\big) = \sigma(2\,\mathrm{pre}_j) \quad \Longrightarrow \quad D_j = \frac{2\,\mathrm{pre}_j}{\ln 2} \approx 2.885\,\mathrm{pre}_j \tag{2}$$

where $D_j = A_j^+ - A_j^-$ is the accumulator difference.

1.2 How Accumulators Are Built

Each weight w_{jk} maps to a byte strength $s_{jk} \in [0, 255]$ plus a sign. For weight $w_{jk} > 0$: pattern $h_k^+ \to h_j^+$ with strength s, and $h_k^- \to h_j^-$ with strength s. For $w_{jk} < 0$: cross-wired ($h_k^+ \to h_j^-$ and $h_k^- \to h_j^+$).

Note on negative weight propagation. Only the softmax output $P(h_j^+)$ propagates to the next layer—h_j^- is consumed within the binary ES and does not carry forward independently. This means a negative weight's contribution ($h_k^+ \to h_j^-$) affects $P(h_j^+)$ via the softmax denominator but does not create a persistent h_j^- signal that chains to further layers. Negative weight information is absorbed at each layer boundary. For recurrent patterns spanning multiple timesteps, this is significant: the h^- side of the ES acts as a local contrast signal, not a propagating one. We carry negative weights in the current implementation because discarding them would lose half the model.

If the source activation is the probability $P(h_k^+)$ from previous softmax, and $h_k = P(h_k^+) - P(h_k^-)$:

$$D_j = \sum_k \text{sign}(w_{jk}) \cdot s_{jk} \cdot h_k \tag{3}$$

For a linear mapping $s_{jk} = \text{round}(\alpha|w_{jk}|)$:

$$D_j \approx \alpha \sum_k w_{jk} \cdot h_k = \alpha(\text{pre}_j - b_j) \tag{4}$$

The matching condition $D_j \approx 2.885\,\text{pre}_j$ requires $\alpha \approx 2.885$. But at $\alpha = 2.885$, the maximum strength is $\text{round}(2.885 \times 4.08) = 12$, using only 5% of the $[0, 255]$ range. Most weights map to 0.

Using higher α sharpens the effective softmax. With temperature compensation (dividing by α before tanh), we can isolate quantization error from temperature distortion.

2 Current Mapping: Why It Fails

The original SN export uses $s = \text{round}(2|w|)$ with threshold $s \geq 1$ ($\alpha = 2$, close to the matching value). This drops 96.3% of weights.

Strength	W_x	W_h	W_y	Total
$s = 0$ (dropped)	98.4%	93.9%	95.5%	96.3%
$s \geq 1$ (exported)	1.6%	6.1%	4.5%	3.7%

Table 1: Weight survival at $\alpha = 2$. 96.3% map to zero.

A UM runner with these patterns gives **7.96 bpc** (near uniform random).

All experiments in this paper use the saturated model (`sat_model.bin`, 128 hidden, 0.079 bpc exact on 1024 bytes of enwik8).

3 Layer Isolation

Quantizing each layer to $[0, 255]$ with its own α (others remain exact float):

W_h **is the bottleneck.** The recurrent layer's quantization errors feed back into themselves, amplifying at each timestep.

4 Linear Mapping with Temperature Compensation

With temperature-compensated linear mapping ($s = \text{round}(\alpha|w|)$, then divide accumulated D_j by α before tanh), varying α:

Without the $[0, 255]$ clip (using wider integers):

Layer quantized	bpc
None (exact)	0.079
W_x only	0.125
$\mathbf{W_h}$ **only**	**0.803**
W_y only	0.080
$W_x + W_h$	1.494
All	2.131

Table 2: Layer isolation. W_h quantization alone accounts for most of the error. W_y quantization is essentially free. Error compounds: $W_x + W_h$ together is worse than the sum of individual effects.

α	Max strength	bpc	W_h dropped
2	8	5.189	93.9%
62.5	255	2.161	10.6%
128	255*	1.553	5.3%

Table 3: Temperature-compensated linear mapping. *Clipped at 255. Even at $\alpha = 128$, bpc is 1.55.

5 Non-linear Mapping: Log-Scale

Since most weights are small ($|w| \approx 0.08$) with a few large outliers ($|w|$ up to 4.08), a log-scale mapping concentrates resolution where it matters:

$$s = \text{round}\left(255 \cdot \frac{\log(1 + c|w|)}{\log(1 + c \cdot w_{\max})}\right) \tag{5}$$

Higher c gives more resolution to small weights. At $c = 1000$ with $w_{\max} = 1.393$ (for W_h): a weight of 0.08 maps to $s = 155$, with resolution $ds/dw = 435$ (one byte step = 0.0023). Per-weight resolution is *excellent*.

5.1 Results: Chaotic Sensitivity

Per-layer log-scale ($c_{W_x} = 1000$, $c_{W_y} = 1000$), sweeping c_{W_h}:

The excellent per-weight resolution (0.002 per byte step) does not translate into stable reconstruction. The reason is **recurrent error amplification**: quantization errors on W_h weights feed back through the hidden state at every timestep. Over 1023 timesteps, even small systematic biases in rounding can amplify chaotically.

5.2 Pre-activation Statistics

$$\text{RMS}(\text{pre}_j) = 1.80, \quad \max|\text{pre}_j| = 11.07, \quad \text{std}(\text{pre}_j) = 1.79$$

Single-step quantization noise (128 W_h weights, resolution $\delta \approx 0.002$):

$$\sigma_{\text{noise}} = \sqrt{128} \cdot \frac{\delta}{2} \approx 0.011$$

This is 0.6% of RMS(pre_j)—apparently negligible. But after T recurrent steps, the error can grow exponentially if the spectral radius of W_h exceeds 1. (For this model, the largest W_h eigenvalue is not computed here, but the empirical instability confirms amplification.)

α	bpc	Notes
128	1.549	
256	1.415	
1024	0.522	
4096	0.160	≈12 bits per weight

Table 4: No-clip quantization. Convergence requires $\alpha \approx 4096$ (≈12 bits).

c_{W_h}	bpc	c_{W_h}	bpc	c_{W_h}	bpc
475	1.111	500	**0.093**	525	0.873
480	0.198	505	0.849	530	0.552
485	0.391	510	0.519	535	1.072
490	0.731	515	0.859	540	0.919
495	0.795	520	1.662	675	**0.088**

Table 5: Fine sweep of c_{W_h}. Results are *chaotic*: 0.093 at $c = 500$, but 0.849 at $c = 505$. The sweet spots at 500 and 675 are isolated, not stable optima.

6 Synthesis

6.1 What Works

1. **Exact float (doubled-E)**: 0.000% bpc difference. Mathematically exact.
2. **≈12-bit quantization**: 0.16 bpc at $\alpha = 4096$ (no clip). Stable, monotonic improvement with resolution.
3. **Lucky log-scale**: 0.088 bpc at isolated c values. Not stable.

6.2 What Doesn't Work

1. **8-bit linear**: 1.45–2.16 bpc. The $[0, 255]$ range cannot span the weight dynamic range (4000:1) and recurrent errors compound.
2. **8-bit log-scale with arbitrary** c: Chaotic results, 0.09–2.2 bpc depending on c. Per-weight resolution is adequate but recurrent amplification creates unpredictable behavior.

6.3 The Fundamental Issue

The problem is not resolution per weight—8 bits give adequate per-weight precision with the right mapping. The problem is **recurrent error amplification**: any quantization error in W_h feeds back through the hidden state, accumulating over 1023 timesteps. This makes the reconstruction quality a chaotic function of the mapping parameters.

However, the RNN was trained with BPTT truncated at 50 steps. The weights were never optimized to propagate information faithfully over 1024 timesteps—only over 50. Error accumulation beyond step 50 is therefore an *artifact of our isomorphic mapping*, not a property of the learned model. The RNN's actual temporal patterns have effective depth ≤ 50; the "chaotic" sensitivity at different c values (Table 5) reflects which rounding choices happen to not blow up over the 974 unoptimized steps. The "lucky" values of c are those where quantization errors happen to cancel or stay bounded beyond the training horizon.

This reframes the problem: faithful reconstruction does not require preserving all 1024 recurrent steps, only the ≤ 50 that carry learned structure. The remaining steps are a free-running dynamical system that the RNN never learned to control.

6.4 W_h as the Recurrent Information Channel

That W_h is the bottleneck is exactly what we should expect. The recurrent weights are the information channel for patterns that span multiple timesteps—the "skip connections through the time dimension." The strength calibration (sn-visibility-sat.pdf) showed the RNN chain strength / UM strength ratio drops from 0.73 (bigrams, 1 recurrent step) to 0.31 (order 11, 10 steps). This monotonic decrease reflects the accumulating information loss as patterns pass through more W_h steps.

With float weights, W_h has $128 \times 128 \times 32 = 524,288$ bits of capacity. At 8 bits per weight, this drops to 131,072 bits. The longer recurrent patterns—exactly those that distinguish this RNN from a bigram model—require the full float channel to propagate faithfully.

6.5 Implications for SN

The approach is always to use the concrete SN representation in $[0, 255]$. Every other decision is downstream of that one. The bpc loss from 8-bit quantization is not a crisis—it is a *signal* about what information is flowing through the float weights that we have not yet captured in the pattern inventory.

In this case, the signal points clearly at **skip connections through the time dimension**. The recurrent W_h weights carry temporal context that compounds over 1023 timesteps. The chaotic sensitivity (Table 5) tells us that specific rounding choices interact with these skip patterns in complex ways. The "lucky" values of c happen to preserve the particular rounding relationships that the longest temporal chains need.

This loss will decrease as interpretation proceeds. When we understand *which* temporal patterns the recurrent weights encode and represent them explicitly in SN (rather than as opaque weight-to-weight chains), the sensitivity to rounding will diminish because the patterns will be stated directly rather than inferred through a chaotic recurrent channel.

6.6 Bayesian View: $Q = \lambda$

The unification paper (20260131_4) establishes $Q = \lambda$: the quotient over microstates equals the luck. Here the microstates are *positions in the dataset.* We derive what the UM forward pass does in these terms.

Setup. The dataset has N positions ($N = 1024$). At position t, the model observes input x_t and predicts x_{t+1}. Each position is a microstate. The *macrostates* are the input and output byte spaces—these are what is observable. The hidden layer is internal structure; when interpreting the UM, we are free to factor the time dimension in different ways, growing an architecture appropriate to the actual patterns.

Input quotient. The event "the input is x_t" partitions the N positions into those where byte t equals x_t and those where it does not. If x_t has frequency $f(x_t)$ in the dataset:

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

The quotient equals the luck of the input event. (The notation λ for luck is felicitous: it also suggests the *wavelength* of occurrences—how far apart instances of the event are spaced in the dataset. An event with $\lambda = 10$ occurs on average every 10 positions.)

Hidden layer quotient. Patterns from the input event fire, accumulating support on hidden events. For binary ES $\{h_j^+, h_j^-\}$ with accumulated difference $D_j = A_j^+ - A_j^-$:

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

Each hidden neuron's softmax is a quotient operation on the remaining consistent positions. The positions where h_j would be positive (in the RNN sense) are separated from those where it would be negative. The product $\prod_j Q_j$ is the cumulative quotient through the hidden layer.

Output quotient and bpc. Output patterns accumulate on byte events; softmax gives $P(x_{t+1} \mid x_t, \mathbf{h})$. The output quotient is $Q_{\text{out}} = 1/P(x_{t+1}) = \lambda(x_{t+1})$. Then:

$$\text{bpc} = \frac{1}{N}\sum_{t=1}^{N} \log_2 Q_{\text{out}}(t) = \frac{1}{N}\sum_{t=1}^{N} \Lambda(x_{t+1})$$

The bpc *is* the average log-luck per position. The export gap $\Delta\text{bpc} = \text{bpc}_{\text{quant}} - \text{bpc}_{\text{float}}$ measures how much additional luck the quantized model needs—how many more positions it fails to distinguish.

Quotient over pattern space. The same framework applies in the other direction: the actual sequences in the dataset determine which patterns can be learned. A bigram ab occurring k times in N positions has frequency k/N, luck $\lambda = N/k$, log-luck $\Lambda = \log_2(N/k)$. The SN strength (as binary log-support) should reflect this luck. The dataset performs a quotient on the space of all possible patterns—only those patterns whose microstates (the positions where they occur) are sufficiently numerous survive as learnable structure.

What quantization loses. Each pattern strength s contributes to the accumulator difference D_j, which determines Q_j. When we round s to an integer, we perturb D_j, which perturbs $Q_j = 1/\sigma(D_j \ln 2)$. For the recurrent layer, the perturbed Q_j feeds back as input to the next timestep. The cumulative quotient over T steps is $Q_T = \prod_{t=1}^{T} Q_j(t)$; a small perturbation in any single Q_j propagates multiplicatively through all subsequent steps. This is why W_h quantization dominates the export gap (Table 2): it is the only layer whose quotient errors compound.

7 UM Pattern Chains: Structure and Taxonomy

7.1 The Input-to-Output Constraint

In the UM, all patterns ultimately chain from input bytes to output bytes:

$$i_{t_1} \to i_{t_2} \to \cdots \to i_{t_k} \to o_y$$

where $i_{t_1}, \ldots, i_{t_k}$ are input events at positions $t_1 < t_2 < \cdots < t_k$ (separated by arbitrary gaps) and o_y is an output byte prediction. The hidden layer events $h_j^{\pm}$ are intermediaries that factor these chains, but they are not observable—only input and output bytes are.

This is a structural constraint on what the UM can represent: every pattern chain must begin at observed inputs and terminate at predicted outputs. The hidden events serve as compressed representations of input subsequences, but the chains they participate in can always (in principle) be unrolled into direct input-to-output form.

7.2 Data-Terms

The microstate view (Section 6.6) shows that whatever computation the RNN performs internally, its UM representation can only operate via *data-terms*: probabilistic syllogisms over events that actually occur in the dataset. A pattern $i_a \to h_j^+ \to o_b$ with strength s is a data-term asserting "when input a is observed, output b receives s bits of log-support, mediated by hidden state j." The strength s is grounded in the dataset frequency (the luck λ) of positions where this co-occurrence holds.

This means the UM does not need to replicate the RNN's internal dynamics. It needs to capture the same probabilistic relationships between observable events—the same data-terms. The recurrent W_h weights encode temporal data-terms (input subsequences predicting outputs), and these are exactly the patterns we seek to make visible in SN.

7.3 Model Taxonomy

We use the following names for the models in this track of work:

sat-rnn The saturated RNN. 128 hidden units, tanh activation, BPTT-50, 0.079 bpc on 1024 bytes. The trained model (`sat_model.bin`).

doubled-E The exact float doubled-E UM. Isomorphic to sat-rnn via $\tanh(x) = 2\sigma(2x) - 1$. Float weights, 0.079 bpc (exact match).

sn-quant The $[0, 255]$ SN-quantized UM. Integer pattern strengths derived from sat-rnn weights. Variable bpc depending on mapping (0.09–2.1).

ngram-um The direct n-gram counting UM. Builds patterns from dataset frequencies. 0.081 bpc at order 11.

pattern-chain Explicit input-to-output chains $i_{t_1} \to \cdots \to o_y$ from dataset frequencies (see pattern-chain.pdf). 0.067 bpc at order 12, surpassing sat-rnn at order 10.

The goal of this sequence of papers is to converge doubled-E $\to$ sn-quant $\to$ pattern-chain: from exact isomorphism to interpretable, explicit data-terms over observable events.

8 Biases as Priors

Biases (b_h, b_y) function as *priors* for each binary ES: before any input-dependent patterns fire, the bias sets the default balance between h_j^+ and h_j^-. A positive b_j biases toward h_j^+; negative toward h_j^-.

These are now exported in the SN pattern inventory as "always-on" patterns from a constant-support source event `"bias"` (support 1). The bias for neuron j becomes a pattern `"bias"` $\rightarrow$ `"h_j+"` (or `"h_j-"`) with appropriate strength. Similarly for output biases.

The full export (`sat_sn_full.c`) produces 3,048 patterns: 522 W_x + 1,006 W_h + 1,471 W_y + 26 b_h + 23 b_y. Total bias magnitudes: $\sum |b_h| = 20.4$ (mean 0.16), $\sum |b_y| = 41.0$ (mean 0.16)—comparable to individual weights.

Reproducibility

```
gcc -O2 -o byte_weight_test byte_weight_test.c -lm
gcc -O2 -o byte_weight_test2 byte_weight_test2.c -lm
gcc -O2 -o byte_weight_test3 byte_weight_test3.c -lm
gcc -O2 -o byte_weight_test4 byte_weight_test4.c -lm
gcc -O2 -o byte_weight_test5 byte_weight_test5.c -lm
./byte_weight_test sat_model.bin enwik_1024.txt
./byte_weight_test2 sat_model.bin enwik_1024.txt
./byte_weight_test3 sat_model.bin enwik_1024.txt
./byte_weight_test4 sat_model.bin enwik_1024.txt
./byte_weight_test5 sat_model.bin enwik_1024.txt
```