The Hidden Quotient

Claude and MJC

8 February 2026

Abstract

The pattern-chain UM builds predictive patterns directly from data, bypassing the RNN's hidden layer. But the hidden layer exists for a reason: it compresses 6,180 data-term patterns into 3,048 SN patterns through 128 neurons. We interpret this compression through the $Q = \lambda$ framework: each hidden neuron encodes a *quotient*—a backward-looking summary of "things that happened" (from I) crossed with a forward-looking prediction of "things expected to happen" (from O). The W_h highway is the working memory that persists these quotients across timesteps, and tanh saturation makes the memory binary and stable. Hidden events become interpretable exactly when we map them onto compound patterns that mention only input and output events—the skip-patterns discovered by the backward trie.

1 Two Directions of Quotient

In the $Q = \lambda$ framework (export-gap.pdf, Section 6.6), every event's strength is grounded in dataset positions—the microstates where the event occurs. A *quotient* is a ratio of position counts: it measures how much more (or less) likely an event is given some condition.

The hidden state h_t sits at the intersection of two quotient directions:

Backward quotient. Given the input history $(x_0, x_1, \ldots, x_t)$, how much evidence does each hidden neuron accumulate? This is $P(h_j \mid \text{past})$: the quotient of positions where neuron j is active, given the observed inputs. These are "things that happened."

Forward quotient. Given hidden state h_t, what output bytes are predicted? This is $P(y \mid h_j)$: the quotient of positions where output y follows activation of neuron j. These are "things expected to happen."

The full prediction factors as:

$$P(y \mid x_0, \ldots, x_t) = \sum_j P(y \mid h_j) \cdot P(h_j \mid x_0, \ldots, x_t)$$

The hidden layer is the *bottleneck* where backward quotients (from I) meet forward quotients (to O). Each neuron h_j is a feature that is simultaneously a summary of input history and a predictor of output.

2 The Highway as Working Memory

The recurrent weights W_h implement a *highway*: information placed into the hidden state at timestep t can persist to timestep $t + d$, with decay governed by the eigenstructure of W_h. The satrnn's W_h has spectral radius 2.52 (export-gap.pdf), meaning some directions are amplified rather than decayed.

In the quotient view, W_h carries backward quotients across time. When a neuron h_j is activated by an input pattern at time t, the highway preserves this activation so that it can contribute to forward quotients at time $t + d$. This is exactly what the greedy skip-k-gram discovers: the byte at offset 8 carries information that persists through W_h and contributes to prediction 8 steps later.

The highway implements dimensional reduction. The full backward-looking quotient space is I^t—all possible input histories. The hidden state compresses this into $\mathbb{R}^{128}$. The skip-k-gram analysis (pattern-chain.pdf, Section 7.5) shows that 4 well-chosen offsets capture most of the predictive information: the hidden state need only preserve these structurally important positions.

3 Saturation and Persistent Memory

The sat-rnn uses tanh activation, which saturates to ± 1 for large inputs. After 4000 epochs of training on 1024 bytes, the activations are deeply saturated: most neurons are near ± 1 at each timestep.

Saturation has a specific interpretation in the quotient framework:

1. A saturated neuron encodes a *binary* quotient: the backward quotient is either "strongly present" $(+1)$ or "strongly absent" (-1). This is a 1-bit summary of input history.

2. Binary quotients are *persistent*. When $h_j \approx +1$ and $W_{h,jj}$ is positive, $\tanh(W_{h,jj} \cdot 1 + \ldots) \approx +1$ again—the state self-sustains. This is the mechanism by which the highway becomes a stable working memory rather than a leaky buffer.

3. The 128 saturated neurons encode up to 2^{128} distinct binary states, but the actual state space is much smaller: the dataset constrains which combinations occur. The number of distinct hidden states is bounded by the dataset size (1024 positions), and in practice many positions share similar hidden states.

Saturation thus explains *how* the RNN makes its working memory persistent: by driving activations to the rails, it converts analog quotients into digital ones that resist decay.

4 Hidden Events as Compound Patterns

A hidden event "neuron h_j is active at time t" is not directly interpretable—it refers to an internal coordinate, not to an observable byte. But we can map it onto compound patterns over I and O:

1. **Backward mapping.** For each neuron j, find which input patterns cause $h_j \approx +1$. These are the backward quotients: subsets of I^k that activate neuron j. The backward trie (pattern-chain.pdf, Section 7.1) discovers exactly these patterns, ranked by MI.

2. **Forward mapping.** For each neuron j, find which output bytes are predicted when $h_j \approx +1$. These are the forward quotients: the conditional distribution $P(y \mid h_j = +1)$.

3. **Compound pattern.** The conjunction of backward and forward mappings gives a compound pattern: "when input pattern A was observed (backward), output byte b is predicted with strength s (forward)." This is a data-term (pattern-chain.pdf, Section 6) that happens to pass through a hidden neuron.

The hidden layer becomes interpretable exactly when every hidden event is resolved into compound patterns that mention only I and O events. The hidden neuron is then understood as a *relay*: a compressed encoding of the input-output patterns it mediates.

5 Finding the Embedding

The question "why does the RNN learn *this particular* hidden representation?" becomes: why does gradient descent converge to this particular set of backward-forward quotient pairs?

The answer has two parts:

1. **The quotient pairs must be stable.** A hidden neuron is useful only if its backward quotient (activation pattern) is consistently correlated with its forward quotient (prediction). Unstable correlations—where the same input pattern sometimes activates h_j and sometimes doesn't—waste capacity. The saturated RNN enforces stability by driving activations to ± 1.

2. **The quotient pairs must be complementary.** 128 neurons must cover the predictive information in 6,180 data-term patterns. Each neuron should encode a backward quotient that is *not already captured* by other neurons—the same principle as the greedy skip-k-gram selector choosing offset 8 before offset 2 because it carries complementary MI.

Finding the embedding is therefore finding a set of 128 stable, complementary correlations between backward-looking quotients (over the I event space) and forward-looking quotients (over the O event space). BPTT discovers these correlations by gradient descent; the backward trie discovers the same information by exhaustive enumeration.

6 Towards Weight Construction

If we can identify the compound patterns that each hidden neuron mediates, we can in principle *construct* the RNN's weights from a forward pass of the UM:

1. Build the backward trie from data (as in pattern-chain.pdf).

2. Select 128 compound patterns (backward-forward quotient pairs) that are stable and complementary.

3. Set W_x to map input bytes to the selected backward quotients.

4. Set W_h to persist the selected quotients across timesteps.

5. Set W_y to map the selected quotients to output predictions.

This would close the loop: instead of training the RNN by gradient descent and then interpreting the result, we would construct the RNN directly from the UM's pattern inventory. The RNN's weights would be *derived* from the data-terms, with the hidden layer serving as a specific compression of the pattern-chain UM into a fixed-width recurrent architecture.

The greedy skip-k-gram result suggests this is feasible: 4 well-chosen offsets (712 patterns) nearly match 12 contiguous bytes (6,180 patterns). The RNN's 128 hidden neurons may be encoding something structurally similar to 128 carefully chosen quotient pairs over the backward trie.

Open Questions

1. Can we empirically verify the backward-forward decomposition by inspecting the sat-rnn's actual neuron activations against the backward trie's MI rankings?

2. What is the minimum number of hidden neurons needed to capture the greedy skip-8-gram's 0.043 bpc? (The skip-8 uses 834 patterns; can 64 neurons suffice?)

3. Does the weight construction (Section 6) produce an RNN that matches the trained RNN's bpc, or does gradient descent find something fundamentally different?

Reproducibility

This paper is analytical; it builds on results from:

```
pattern-chain.pdf (Sections 5, 7)
export-gap.pdf (Sections 6.5, 6.6)
```