Q1 Protocol C: The f32 Quotient

Claude and MJC

11 February 2026

Abstract

The superset UM u_{sup} lives in $\mathbb{R}^{\text{dim}}$ where $\text{dim} = 128 \times 256 + 128^2 + 256 \times 128 + 128 + 256 =$ 82,048. The RNN computes in f32, a finite quotient of $\mathbb{R}$. We trace what this quotient does to pattern strengths ω across the model taxonomy (sat-rnn, doubled-E, constructed models), predict via information flow analysis how f32 precision affects long witness chains under BPTT, describe the resulting learning dynamics under Adam, and measure the entropy and variance of learning efficiency.

1 The f32 Quotient

Definition 1 (The quotient $\mathbb{R} \to$ f32)**.** *Let $q\colon \mathbb{R} \to$ f32 be the rounding map that sends each real number to its nearest IEEE 754 single-precision representation. This is a quotient: q identifies all reals in the interval $[x - \frac{1}{2}ulp(x),\ x + \frac{1}{2}ulp(x))$ with the same f32 value x.*

The key properties:

- ***Finitizing**: f32 has 2^{32} values. The quotient replaces the uncountable $\mathbb{R}$ with a finite set, making the model computable.*
- ***Non-uniform**: ulp (unit in the last place) depends on magnitude. Near zero, ulp $\approx 1.4 \times 10^{-45}$. Near 1, ulp $\approx 1.2 \times 10^{-7}$. Near saturation ($\tanh \approx \pm 1$), ulp $\approx 6 \times 10^{-8}$.*
- ***Compositional error**: each f32 operation introduces relative error $\leq 2^{-24} \approx 6 \times 10^{-8}$. A chain of k operations accumulates error $\sim k \cdot 2^{-24}$ (worst case) or $\sim \sqrt{k} \cdot 2^{-24}$ (random walk).*

1.1 Bit decomposition of f32

An f32 value has 32 bits: 1 sign bit s, 8 exponent bits $e_7 \ldots e_0$, and 23 mantissa bits $m_1 \ldots m_{23}$. The value is:

$$v = (-1)^s \times 2^{E-127} \times \left(1 + \sum_{i=1}^{23} m_i \cdot 2^{-i}\right)$$

where $E = \sum_{k=0}^{7} e_k \cdot 2^k$ is the exponent.

Each mantissa bit m_i contributes $(-1)^s \cdot 2^{E-127-i}$ to the value. The contribution depends on the exponent: bit m_1 contributes 2^{E-128}, bit m_{23} contributes 2^{E-150}.

Information channels. Multiply each bit by its index, then quotient out the exponent. What remains is the pure bit pattern: 23 independent binary channels, each carrying at most 1 bit of information. The exponent sets the *scale* (which power of 2 the channels operate at); the mantissa bits carry the *content*.

Now map the input entropy onto these channels. A weight $\omega(p)$ encodes the strength of pattern p. If p participates in k predictions, the weight must encode k pieces of information. These k pieces share 23 mantissa bits. The information capacity per weight is:

$$C_w = 23 \text{ bits (mantissa)} + 8 \text{ bits (exponent)} = 31 \text{ bits}$$

but the exponent and high mantissa bits encode the overall scale (how important this pattern is), while the low mantissa bits encode fine distinctions between similar predictions.

Entropy bleed. As training progresses, the data's entropy flows into the weights. Initially, the mantissa bits are random (from init). After training, the high bits lock in to encode the pattern's sign and gross strength, while the low bits encode residual structure.

The entropy *bleeds* from high bits to low bits: early training sets the exponent and top mantissa bits (coarse pattern identity), later training refines the low mantissa bits (fine strength adjustment). At convergence, the number of "information-carrying" mantissa bits per weight is:

$$b_{\text{eff}}(p) = \log_2 \frac{|\omega(p)|}{\text{ulp}(\omega(p))} = \begin{cases} 23 & \text{if } \omega \text{ is fully utilized} \\ < 23 & \text{if low bits are noise} \end{cases}$$

For the sat-rnn, we can measure b_{eff} for each of the 82,048 weights and determine how much of the f32 capacity is actually used by the learned patterns. The total information content of the trained model is $\sum_p b_{\text{eff}}(p)$ bits, which upper-bounds the number of data patterns the model can represent.

Definition 2 (f32-quotiented UM)**.** *Let $u_{f32} = q(u_{\text{sup}})$ be the UM with all weights rounded to f32. For a pattern p with exact strength $\omega(p) \in \mathbb{R}$, the f32 strength is $\omega_{f32}(p) = q(\omega(p))$.*

The error per pattern is $|\omega(p) - \omega_{f32}(p)| \leq \frac{1}{2} ulp(\omega(p))$, which is negligible for individual patterns. The question is what happens to chains.

2 Effect on Witness Chains

A witness chain of depth d composes $d + 2$ patterns:

$$w(C) = W_x[j_0, b] \cdot \prod_{s=0}^{d-1} (1 - h_{j_s}^2) \cdot W_h[j_{s+1}, j_s] \cdot W_y[y, j_d]$$

In f32, each factor is rounded, and each multiplication introduces rounding error. The chain weight becomes:

$$w_{\text{f32}}(C) = w(C) \cdot \prod_{i=1}^{2d+2} (1 + \epsilon_i) \qquad |\epsilon_i| \leq 2^{-24}$$

2.1 Error growth with depth

The relative error of the chain weight is:

$$\frac{|w_{\text{f32}}(C) - w(C)|}{|w(C)|} \leq (1 + 2^{-24})^{2d+2} - 1 \approx (2d + 2) \cdot 2^{-24}$$

At $d = 50$ (BPTT horizon): relative error $\leq 102 \cdot 2^{-24} \approx 6 \times 10^{-6}$. This is small—f32 precision is not a problem for individual chain weights.

But the backward gradient $g_{t,d}$ is a *sum* over 128^{d+1} chains. The error of the sum depends on cancellation:

- If chains are correlated (same sign): errors add constructively. Relative error $\sim d \cdot 2^{-24}$. Still small.
- If chains cancel (mixed signs): the sum is small relative to the individual terms, and the absolute error dominates. This is catastrophic cancellation.

2.2 Predicting gradient precision via information flow

The saturation gates $(1 - h_j^2)$ determine which chains survive. At each step, a fraction ϕ of neurons are saturated (gate $< \epsilon$). The effective number of active chains at depth d is $\sim (128(1 - \phi))^d$.

From Protocol A, the depth profile shows attribution mass peaking at $d \approx 20$ and remaining substantial to $d = 50$. This means the gradient is NOT dominated by cancellation—the surviving chains are coherent (same sign), which preserves f32 precision.

We predict: f32 precision is sufficient for the backward gradient at all depths $d \leq 50$ because:

1. Saturation kills most chains, keeping the effective count low.
2. Surviving chains are coherent (selected by the same gradient direction), so cancellation is mild.
3. The relative error $(2d + 2) \cdot 2^{-24}$ is $< 10^{-5}$ even at $d = 50$.

2.3 Worked example: entropy bleeding through one BPTT step

Consider position $t = 42$, predicting $y =$ `m` (byte 109). We trace one backward step from $d = 0$ to $d = 1$, decomposing every operation into f32 channels.

Step 0: Output gradient. The output gradient at neuron j is:

$$[g_t]_j = W_y[109, j] - \sum_{o=0}^{255} W_y[o, j] \cdot P_t(o)$$

Consider neuron $j = 7$. Suppose $W_y[109, 7] = 1.847$ and the P-weighted sum is 0.312. Then $[g_t]_7 = 1.535$.

In f32 channels:

- **Sign**: positive $\Rightarrow$ neuron 7 is in the h_7^+ state and this *helps* predict `m`.
- **Exponent**: $1.535 = 1.535 \times 2^0$, so $E = 127$. This is an order-1 gradient—a strong signal.

- **Mantissa**: $1.535/2^0 - 1 = 0.535$, encoded in 23 bits as the fractional part. This precision distinguishes $[g_t]_7 = 1.535$ from 1.536.

The *information content* of $[g_t]_7$: the sign tells us the ES decision (h_7^+, 1 bit). The exponent tells us importance (strong, 8 bits but really ∼3 bits of entropy since most gradients cluster near a few exponent values). The mantissa carries ∼10–15 bits of useful precision (the rest is noise from the softmax sum over 256 terms).

Step 1: Jacobian propagation. The backward gradient propagates one step:

$$[g_{t,1}]_j = \sum_k (1 - h_k(t)^2) \cdot W_h[k, j] \cdot [g_t]_k$$

For neuron $j = 3$, this sums over all 128 source neurons k. Each term has three f32 factors:

Factor	Channel	Entropy	What it encodes
$(1 - h_k^2)$			
sign	always +	0 bits	gate is non-negative
exponent	E of gate	∼3 bits	is k saturated?
mantissa	gate value	∼7 bits	how open is k?
$W_h[k, j]$			
sign	±	1 bit	excitatory or inhibitory
exponent	E of weight	∼3 bits	connection strength class
mantissa	fine strength	∼15 bits	precise strength
$[g_t]_k$			
sign	±	1 bit	does k help or hurt?
exponent	E of grad	∼3 bits	how important is k?
mantissa	fine grad	∼12 bits	precise gradient

The product $(1 - h_k^2) \cdot W_h[k, j] \cdot [g_t]_k$ is one f32 multiply chain. The entropy flows as:

1. **Sign of product**: $\text{sgn}(W_h[k, j]) \cdot \text{sgn}([g_t]_k)$. Two bits in, one bit out. This determines whether neuron k's signal helps or hurts neuron j—the *direction* of the pattern chain.

2. **Exponent of product**: $E_{\text{gate}} + E_{\text{weight}} + E_{\text{grad}} - 2 \times 127$. Three exponents in, one out. If the gate exponent is small (neuron saturated, $E_{\text{gate}} \ll 127$), the product exponent drops below the noise floor and the entire term vanishes. *This is where entropy bleeds*: the gate's exponent channel kills the information from the weight and gradient channels. Saturation is exponent-level filtering.

3. **Mantissa of product**: the product of three mantissas, each with ∼23 significant bits, gives a result with 23 significant bits (f32 rounds). Information from the three input mantissas is mixed and truncated. The fine-grained precision of the weight and gradient are partially lost.

The sum over k. After computing all 128 terms, we sum: $[g_{t,1}]_3 = \sum_{k=0}^{127} \text{term}_k$.

The sum introduces a new level of entropy bleeding:

- Terms with different signs partially cancel. Each cancellation destroys mantissa bits: if two terms of magnitude ∼1 cancel to give a result of magnitude ∼0.01, two decimal digits (about 7 binary bits) of mantissa precision are lost.

- Terms with small exponents (killed by saturation gates) contribute nothing to the sum—their entropy has already bled out at the exponent level.
- The surviving terms' exponents must be within ∼23 powers of 2 of each other, or the smaller terms are lost to f32 rounding of the partial sum. This is a second exponent-level filter.

In the sat-rnn, Protocol A showed that the gradient norm at $d = 1$ is ∼0.23× the $d = 0$ norm. This means: of the 128 terms in the sum, the coherent part (same-sign terms surviving saturation) retains 23% of the original signal. The remaining 77% has bled out through saturation gates (exponent channel) and cancellation (mantissa channel).

The cascade. At depth $d = 2$, the same process repeats: 128 terms, each a product of gate × weight × $g_{t,1}$. But now $g_{t,1}$ has already lost entropy from depth 1. The mantissa of $g_{t,1}$ carries ∼15 bits (down from 23, due to the sum). After another multiply-and-sum, $g_{t,2}$ has ∼12 bits.

The exponent channel decays more slowly—it drops by ∼1–2 per depth step (the geometric mean of the saturation gates and weight magnitudes). This is why the exponent is the most stable channel: it survives the deepest into BPTT, carrying the coarse "is there signal from this depth?" information even when the mantissa has been destroyed by cancellation.

At depth $d \approx 20$ (the attribution mass peak from Protocol A), the exponent still carries ∼3 useful bits, but the mantissa may be down to ∼5–8 bits. At $d = 50$, the exponent determines whether there is any signal at all; the mantissa is almost entirely noise.

3 Effect on Learning via Adam

The Adam optimizer updates each weight w as:

$$w \leftarrow w - \eta \cdot \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon}$$

where $\hat{m}$ is the bias-corrected first moment (gradient) and $\hat{v}$ is the bias-corrected second moment (gradient variance).

3.1 Learning as pattern strength update

In UM terms, each weight is a pattern strength. The Adam update is:

$$\omega(p) \leftarrow \omega(p) + \Delta\omega(p)$$

where $\Delta\omega(p)$ depends on the gradient of the loss with respect to pattern p's strength. For a W_x pattern (b, j):

$$\frac{\partial \mathcal{L}}{\partial W_x[j, b]} = -\sum_{t:x_t=b} [g_{t,0}]_j$$

This is the sum of backward gradients at all positions where byte b is the input, projected onto neuron j. Adam normalizes this by its running variance, giving an effective learning rate per pattern.

3.2 f32 effect on learning

The f32 quotient affects learning in two ways:

1. **Gradient precision**: the gradient $\partial\mathcal{L}/\partial\omega(p)$ has f32 rounding error as analyzed above. For short chains ($d \leq 5$), this is negligible. For long chains ($d > 20$), the error may exceed the true gradient for weak patterns.

2. **Update precision**: the weight update $\Delta\omega(p)$ must exceed $\text{ulp}(\omega(p))$ to have any effect. For $|\omega(p)| \sim 1$, ulp $\approx 10^{-7}$. With learning rate $\eta \sim 10^{-3}$ and gradient $\sim 10^{-2}$, the update is $\sim 10^{-5}$, well above ulp. But for large weights ($|\omega| > 10$) or small gradients, stagnation can occur.

4 The Epsilon Field: Entropy at Initialization

Before any training, the weights are drawn from some initialization distribution (e.g. Xavier: $w \sim \mathcal{N}(0, 1/\text{fan\_in})$). These random weights already define a UM u_{init} with 82,048 patterns, each with a random strength $\omega(p) \sim \mathcal{N}(0, \sigma^2)$.

4.1 Measurable entropy at init

The randomly initialized UM makes predictions. At each position t:

$$H_{\text{init}}(t) = -\sum_{y} P_{\text{init}}(y \mid \text{context}_t) \log_2 P_{\text{init}}(y \mid \text{context}_t)$$

Since the weights are random, P_{init} is approximately uniform over the output alphabet, giving $H_{\text{init}} \approx \log_2 52 \approx 5.7$ bits (for 52 distinct bytes). The bpc at init is approximately $\log_2 256 = 8$ bpc for the full byte alphabet.

But H_{init} is not exactly $\log_2 256$. The random weights break symmetry: some output bytes are randomly favored, some input-output pairs are randomly correlated. The deviation from maximum entropy is:

$$\Delta H_{\text{init}} = \log_2 256 - H_{\text{init}}$$

This is the information content of the random initialization—the "free" structure that training does not need to learn.

4.2 Lottery tickets as accidental patterns

A *lottery ticket* in this framework is a pattern p in u_{init} whose random strength $\omega_{\text{init}}(p)$ happens to have the correct sign and significant magnitude relative to the data-derived strength $\omega_{\text{data}}(p)$:

$$\text{sign}(\omega_{\text{init}}(p)) = \text{sign}(\omega_{\text{data}}(p)) \quad \text{and} \quad |\omega_{\text{init}}(p)| > \tau$$

By chance, roughly half the patterns have the correct sign (for symmetric init). The fraction with $|\omega_{\text{init}}| > \tau$ depends on σ/τ. A "winning ticket" is a subnetwork where enough patterns are accidentally correct that the model already compresses better than chance.

The conditions for a lottery ticket are now precise:

1. The LPP must define event spaces that contain the data's patterns (otherwise no ticket exists regardless of init).

2. The init distribution must have enough variance that $P(|\omega_{\text{init}}(p)| > \tau)$ is nontrivial for the relevant patterns.

3. The number of relevant patterns must be small enough that the probability of "all correct signs" is nontrivial—this is the sparsity condition from Q1.

From Protocol A, the median prediction uses only 15 patterns at $\tau = 0.01$. The probability that all 15 have the correct sign by chance is $2^{-15} \approx 3 \times 10^{-5}$. Over 1023 positions, the expected number of positions with a "perfect lottery ticket" is ~ 0.03. Lottery tickets for individual predictions are rare; lottery tickets for the entire model are exponentially rare.

This quantifies why training is necessary: the f32 epsilon field at initialization has the right structure (event spaces, pattern space) but the wrong strengths. Learning is the process of replacing ω_{init} with ω_{data}.

5 Measurements

5.1 Pattern strength distribution in f32

For each model in the taxonomy (sat-rnn, doubled-E, constructed models), compute:

- Distribution of $\omega(p)$ values.
- Distribution of $\text{ulp}(\omega(p))$.
- Fraction of patterns where $\text{ulp} > |\Delta\omega|$ (stagnation threshold).

Hidden state signs as event space embedding. At each position t, the sign vector $\text{sgn}(h_t) \in \{-1, +1\}^{128}$ determines which binary ES event occurred at each neuron. This embeds the hidden state into the vertices of the 128-dimensional hypercube $\{-1, +1\}^{128}$, which has 2^{128} vertices. In practice, the sat-rnn on 1024 bytes visits at most 1024 distinct sign vectors—an exponentially sparse subset.

Each weight $W_x[j, b]$ participates in patterns for all positions where $x_t = b$. At those positions, neuron j may be in state h_j^+ or h_j^-, giving two patterns with the same weight but opposite signs. The single f32 value encodes both patterns simultaneously—it is a superposition over the binary ES.

Factoring into true event spaces. To see what the f32 bits actually encode, factor each weight matrix into the event spaces it connects:

- $W_x[j, b]$: connects $E_{\text{in}} \times E_{h_j}$. This is a 52×2 matrix (52 observed input bytes, 2 hidden events per neuron). Each weight carries information about one $(b, h_j^\pm)$ pair.
- $W_h[k, j]$: connects $E_{h_j} \times E_{h_k}$. Each weight carries information about one $(h_j^\pm, h_k^\pm)$ pair across adjacent timesteps.
- $W_y[y, j]$: connects $E_{h_j} \times E_{\text{out}}$. Each weight carries information about one $(h_j^\pm, y)$ pair.

Now multiply by time: in u_{sup}, the pattern (b, h_j^+, t)—byte b excites neuron j at time t—uses the same weight $W_x[j, b]$ at every position where $x_t = b$. The single f32 value is shared across all these time positions. This is the fundamental bottleneck: the 23 mantissa bits must encode the pattern strength that works for *all* positions where this pattern is active.

Full factorization into u_{sup}. Each f32 weight has three information channels: sign (1 bit), exponent (8 bits), mantissa (23 bits). We factor each separately.

Channel 1: Sign (1 bit per weight). The sign bit is the most stable channel and the most semantically meaningful. It determines whether a pattern is excitatory or inhibitory: $W_h[k, j] > 0$ means "h_j^+ excites h_k^+"; $W_h[k, j] < 0$ means "h_j^+ inhibits h_k^+" (equivalently, "h_j^+ excites h_k^-"). The sign bit encodes which of the two binary ES events the pattern connects to.

Total sign channel: 29,696 used weights $\times$ 1 bit = 29,696 bits. This encodes the *topology* of the UM—which events are connected and in which direction.

Channel 2: Exponent (8 bits per weight). The exponent encodes the order of magnitude of the pattern strength: 2^{E-127}. This is the **most stable channel** during learning. Changing the exponent requires the weight to cross a power-of-2 boundary—doubling or halving its magnitude. Once training establishes whether a pattern is strong ($E \geq 127$, i.e. $|w| \geq 1$) or weak ($E < 127$, i.e. $|w| < 1$), the exponent rarely changes again.

The exponent encodes the pattern's *importance ranking*: which patterns matter at all. In UM terms, this is the first-order structure of ω—the partition of patterns into significant and insignificant.

Total exponent channel: 29,696 $\times$ 8 = 237,568 bits.

Channel 3: Mantissa (23 bits per weight). The mantissa encodes the fine-grained strength within the scale set by the exponent. It is the noisiest channel: every gradient update modifies the mantissa, and it continues to be refined throughout training. The mantissa carries the *residual information*—the precise pattern strength that determines the difference between 0.079 bpc and 0.08 bpc.

Total mantissa channel: 29,696 $\times$ 23 = 682,608 bits.

Entropy flow across channels. Now multiply by the event spaces and time. Each weight serves multiple pattern instances in u_{sup}:

- $W_x[j, b]$: serves $n_b \approx 20$ time positions.
- $W_h[k, j]$: serves 1023 transitions.
- $W_y[y, j]$: serves 1023 predictions.

The information budget per instance, factored by channel:

	Weights	Sign/inst	Exp/inst	Mant/inst
W_x	6,656	1/20	8/20	23/20
		= 0.05	= 0.40	= 1.15
W_h	16,384	1/1023	8/1023	23/1023
		= 0.001	= 0.008	= 0.022
W_y	6,656	1/1023	8/1023	23/1023
		= 0.001	= 0.008	= 0.022

The exponent channel for W_h gives 0.008 bits per transition. This is the budget for encoding *whether this recurrent connection matters at all* at a given timestep. The model cannot make per-position decisions about importance—the exponent is shared across all 1023 transitions. It must

commit: this connection is either always important (large exponent) or never important (small exponent).

The sign channel for W_h is even tighter: 0.001 bits per transition. The excitatory/inhibitory nature of each recurrent connection is fixed for all time. The model cannot reverse a connection's sign at one position without reversing it everywhere.

This is why the sign vector $\text{sgn}(h_t) \in \{-1, +1\}^{128}$ matters: it is the *only* mechanism by which time-varying information enters the binary ES decisions. The weights set the topology (sign channel) and importance (exponent channel) once for all time; the hidden state dynamics modulate *which* patterns are active at each position by flipping neurons between h_j^+ and h_j^-.

Where entropy bleeds. The input data has entropy $H_{\text{data}} \approx 5{,}800$ bits (1023 predictions $\times$ 5.7 bits marginal). The model compresses to $\sim$81 bits (0.079 bpc). The $\sim$5,700 bits of compression must be encoded somewhere. It distributes across channels:

1. **Sign channel** (29.7K bits capacity): encodes the UM topology. Learned early, stable. A small fraction of the 5,700 bits—perhaps $\sim$500 bits (the number of sign decisions that actually matter for the redux's $\sim$1000 patterns).

2. **Exponent channel** (237.6K bits capacity): encodes which patterns are significant. The exponent of each weight determines the pattern's ranking. With $\sim$1000 patterns in the redux, the exponent encodes $\sim$1000 importance decisions at 8 bits each $=\sim$8000 bits of information. This exceeds the compression (5,700 bits), so the exponent channel alone has enough capacity.

3. **Mantissa channel** (682.6K bits capacity): encodes the fine-grained strengths. Massively overcapacity for this task. Most mantissa bits are noise—they encode structure that is never used in the backward attribution chain.

The learning is low-dimensional because the compression lives primarily in the exponent channel, which has 237.6K bits of capacity for $\sim$5,700 bits of compression—a 42:1 ratio. The mantissa channel (120:1 ratio) refines but does not fundamentally contribute. The sign channel (5:1 ratio) is the tightest, but sign decisions are binary and learned first.

5.2 Entropy of the f32 UM

The entropy of the f32-quotiented UM at position t:

$$H_{\text{f32}}(t) = -\sum_{y} P_{\text{f32}}(y \mid \text{context}_t) \log_2 P_{\text{f32}}(y \mid \text{context}_t)$$

Compare to the exact entropy $H_{\text{exact}}(t)$. The difference $H_{\text{f32}} - H_{\text{exact}}$ is the entropy cost of the f32 quotient.

5.3 Variance of learning efficiency

For each pattern p, the learning efficiency is:

$$\eta_{\text{eff}}(p) = \frac{|\Delta\omega_{\text{f32}}(p)|}{|\Delta\omega_{\text{exact}}(p)|}$$

Patterns with $\eta_{\text{eff}} \approx 1$ are learned accurately. Patterns with $\eta_{\text{eff}} \ll 1$ are lost to f32 noise. The variance of η_{eff} across patterns measures how uniformly the f32 quotient affects learning.

5.4 BPTT depth and f32 noise floor

For each depth d, compute:

- Mean gradient norm $\|g_{t,d}\|$ in f32.
- Mean gradient norm in f64 (as ground truth).
- Ratio: fraction of gradient signal surviving f32.
- Depth d^* at which f32 gradient falls below noise floor.

This tells us the effective BPTT depth under f32—which may be shorter than the nominal 50.

6 Discussion

Reproducibility

Tool: `q1_protocol_c.c` in `docs/archive/20260211/`. Models: sat-rnn, constructed models from `archive/20260208-09/`. Data: first 1024 bytes of enwik9.