RAVEL: COLLISION-VERIFIED STATE AUGMENTATION FOR LONG-CONTEXT RECURRENT LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fixed-state recurrent language models enable linear-time processing and constant-state decoding, but repeatedly compressing the prefix into a small state can cause interference between independent associations. We introduce RAVEL, a recurrent architecture with bounded collision-verified state augmentation. RAVEL maintains six fixed direct-address banks alongside the recurrent core, where each slot stores an exact token fingerprint, a support count, and a differentiable successor-value prototype. Reads precede writes, fingerprint mismatches abstain exactly, and a learned residual gate with a delayed horizon protects short-context behavior. The resulting augmentation requires no attention, nearest-neighbor search, relevance ranking, or unbounded token cache. We establish causal equivalence between teacher-forced and autoregressive execution, exact collision abstention, bounded state values and logit corrections, convergence of the running prototype, and an exponential-in-bank-count bound on surviving collisions. Across matched 6.1M, 11.7M, and 23.7M parameter scales, three-seed TinyStories experiments show that RAVEL improves over matched Mamba-3 at context lengths 128–4096 while remaining statistically indistinguishable at length 32, and improves over RWKV-7 at every measured length. A post-hoc TinyShakespeare study reproduces the long-context advantage. Frozen ablations show that one verified bank provides most of the gain, while additional banks offer diminishing collision redundancy. Together, these results demonstrate that RAVEL delivers a consistent and interpretable long-context quality advantage while preserving the bounded-state structure of recurrent language models.

1 INTRODUCTION

Modern sequence models make a consequential choice about where history lives. Transformers retain token-level keys and values, providing content access at quadratic training cost and linearly growing decode memory (Vaswani et al., 2017). State-space and linear recurrent models instead fold the prefix into a fixed-size state, enabling linear training and constant-state decoding (Gu & Dao, 2023; Dao & Gu, 2024; Peng et al., 2025; Lahoti et al., 2026). Their compression is attractive, but independent associations can interfere: a recurrent state that is excellent at smooth dynamics need not preserve every local "key $\rightarrow$ successor" event.

Existing responses span neural caches (Grave et al., 2017), learned test-time memories (Behrouz et al., 2025), static conditional lookup (Cheng et al., 2026), and bounded exact caches beside linear attention (Cui, 2026). These systems establish that an auxiliary state path can recover information a fixed recurrent state loses. They also raise a sharper systems-and-modeling question: *what is the smallest auditable state augmentation that measurably reduces association interference without becoming attention or search?*

We answer with RAVEL, shown in Figure 1. A Mamba-3 core produces base logits. In parallel, the current token addresses J fixed banks. Each addressed slot is accepted only when its stored fingerprint exactly matches the token. Valid values are averaged, projected through the shared language-model head, gated by the hidden state, and activated only after a fixed short-context horizon. After the current prediction, the realized successor token updates each addressed bank by a running mean or

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

replaces a collided slot. The memory is bounded and deterministic: $O(JBd_v)$ state, $O(Jd_v)$ work per token for fixed J, no pairwise token comparisons, and no learned eviction search.

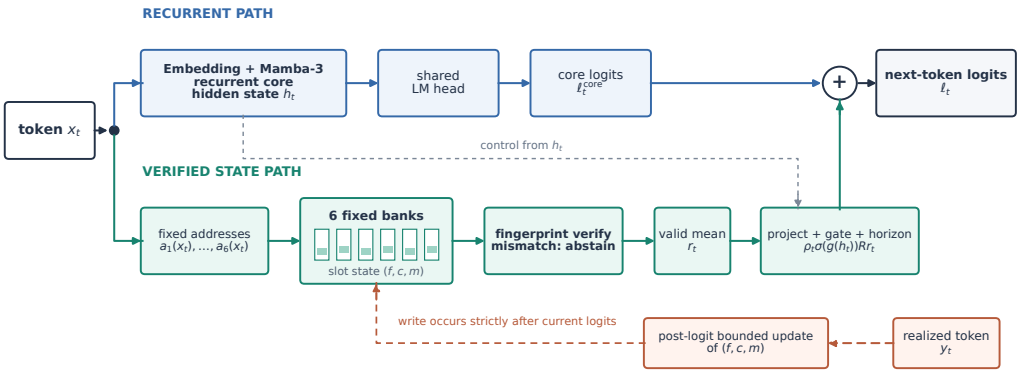


Figure 1: RAVEL augments a Mamba-3 recurrent core with six bounded, direct-address banks. Fingerprint verification turns hash collision into abstention rather than a false read. The realized next token is written only after the current logit, making teacher-forced evaluation and autoregressive deployment causally aligned. A learned scalar gate and fixed horizon add the memory correction without changing L32 behavior.

Our claim is intentionally narrower than "memory solves long context." The model is evaluated under frozen development contracts, not broad downstream test suites. Native full-vs-decode paths for several official baselines did not meet the preregistered fidelity criteria, so reference-runner throughput is reported only as a limitation. Within this scope, the evidence is unusually direct: three matched parameter scales, three seeds, paired deterministic windows, an independently checked operator, OOD transfer, and a bank-count ablation that isolates the mechanism without retraining.

Our contributions are:

- **A verified bounded state augmentation.** Fixed addresses, exact fingerprints, read-before-write ordering, and running successor prototypes add content association without attention, ANN, ranking, or an unbounded cache.
- **A mathematical account of behavior.** We prove causality, exact abstention, boundedness, prototype convergence, and the collision-survival benefit of multiple independent banks.
- **Matched scaling and transfer evidence.** At 6.1M, 11.7M, and 23.7M parameters, RAVEL consistently improves long-context NLL over Mamba-3 and RWKV-7; a post-hoc OOD study shows the same boundary.

2 Related Work

Fixed-state sequence models. Mamba introduced selective state-space dynamics (Gu & Dao, 2023); Mamba-2 connected structured state spaces and attention duality (Dao & Gu, 2024); Mamba-3 adds a more expressive discretized recurrence, complex state, and MIMO formulation (Lahoti et al., 2026). RWKV-7 generalizes the delta rule using vector gates and in-context learning rates (Peng et al., 2025). Gated DeltaNet combines decay and delta-rule writes (Yang et al., 2024); Kimi Delta Attention develops finer-grained finite-state gating and efficient kernels (Kimi Team, 2025). These models compress history into structured continuous state. RAVEL does not replace their dynamics; it adds a typed exact-association path for information that is awkward to compress.

Explicit and test-time memory. Continuous neural caches reuse past hidden states (Grave et al., 2017). Titans learns a neural long-term memory at test time (Behrouz et al., 2025). Engram performs scalable static n-gram lookup (Cheng et al., 2026). HOLA adds a bounded exact KV complement to a delta-rule state and chooses writes from prediction residuals (Cui, 2026). YOCO reduces repeated KV caching while retaining attention access (Sun et al., 2024). RAVEL differs operationally: it is

sequence-local and online, addresses lexical keys deterministically, validates every read with an exact fingerprint, and uses no relevance ranking. We do not claim that this strictly dominates learned or content-addressed memory; rather, it isolates a low-complexity collision-verified primitive.

Capacity and succinctness. Fixed-precision Transformers can be exponentially more succinct than recurrent models for some formal languages (Bergsträßer et al., 2026). That separation cautions against universal expressivity claims for any fixed-state architecture. Our result concerns finite-context empirical association under a bounded external state. Multiple banks resemble sketch redundancy (Cormode & Muthukrishnan, 2005), but fingerprints eliminate false positive reads and the stored objects are differentiable running prototypes rather than frequency counts.

3 RAVEL

Let $x_t \in [V]$ be the current token and $y_t = x_{t+1}$ the realized successor under teacher forcing. A recurrent core produces h_t and base logits

$$h_t = F_\theta(x_{\le t}), \qquad z_t^{\text{core}} = W_o h_t. \tag{1}$$

The memory has J banks and B slots per bank. A fixed map $a_j : [V] \to [B]$ addresses bank j. Slot (j, b) stores a tuple $(f_{j,b}, c_{j,b}, m_{j,b})$: a token fingerprint, a capped count, and a value vector in $\mathbb{R}^{d_v}$.

3.1 Verified read before write

At step t, bank j returns

$$q_{t,j} = \mathbf{1}\{f_{j,a_j(x_t)} = x_t\}, \tag{2}$$

$$r_{t,j} = q_{t,j}\, m_{j,a_j(x_t)}, \qquad r_t = \frac{\sum_{j=1}^{J} r_{t,j}}{\max(1, \sum_{j=1}^{J} q_{t,j})}. \tag{3}$$

Thus a hash collision never silently returns another token's value. After computing the current logit, $v_t = E_v(y_t)$ is committed. A verified match performs

$$c' = \min(c + 1, C), \qquad m' = m + \frac{v_t - m}{c'}, \tag{4}$$

while an empty or collided slot becomes $(x_t, 1, v_t)$. The deployed autoregressive procedure writes the token actually produced at step t before processing step $t + 1$; the training implementation receives y_t in the same operation but does not expose it to the current read.

The memory correction is

$$z_t = z_t^{\text{core}} + \rho_t\, \sigma(g(h_t))\, W_o R r_t, \qquad \rho_t = \text{clip}\left(\frac{t - 64}{32}, 0, 1\right). \tag{5}$$

The fixed horizon makes the memory path identically zero through position 64 and fully active from position 96 onward. It is a declared architectural prior, not a post-hoc evaluation mask.

3.2 Complexity

For fixed J, each token performs J address lookups, exact comparisons, and d_v-dimensional value updates. The additional recurrent state is

$$O(JB(d_v + 1)), \qquad \text{and per-token work is } O(Jd_v). \tag{6}$$

The audited 11.7M model uses $J = 6$, $B = 1024$, $d_v = 64$, and 1,646,592 bytes per sequence in the reference representation. Unlike attention, state does not grow with context; unlike ANN or sparse attention, a query does not compare against stored keys.

4 Why the Mechanism Works

We state the main properties here and give complete proofs in the supplementary material.

Theorem 1 (Causality and read-before-write)**.** *For fixed parameters and address maps, z_t is a function of $(x_{\le t}, y_{<t})$ and is independent of y_t. Updating the deployed state after realizing y_t therefore produces the same state available at step $t+1$ as teacher forcing.*

The result follows because Equation (3) is evaluated from the pre-write state; Equation (4) is applied only after z_t. This property is verified by the causality tests in the released package.

Proposition 1 (Prototype optimality and boundedness)**.** *Suppose a slot receives verified values $v_1, \dots, v_n$ without an intervening collision and the count is uncapped. Then its state is*

$$m_n = \frac{1}{n}\sum_{i=1}^{n} v_i = \arg\min_{m} \sum_{i=1}^{n} \|m - v_i\|_2^2. \tag{7}$$

If $\|E_v(y)\| \le M$, every slot value satisfies $\|m\| \le M$ and the memory-logit correction is bounded by $\|W_o\|\,\|R\|M$.

Thus the memory is not an unconstrained fast-weight optimizer: it is an exact online conditional-prototype estimator with convex updates. Under i.i.d. successors and uninterrupted verified matches, m_n converges almost surely to $\mathbb{E}[E_v(Y) \mid X = x]$.

Theorem 2 (Independent-bank collision survival)**.** *Fix a token x and suppose K distinct interfering token types are written after its last occurrence. Under independent uniform bank maps into B slots, one bank retains a verified value with probability*

$$p_K = \left(1 - \frac{1}{B}\right)^K, \tag{8}$$

and at least one of J banks survives with probability

$$P_{J,K} = 1 - (1 - p_K)^J. \tag{9}$$

The all-bank miss probability decays exponentially in J, while work and state grow linearly. This predicts a large jump from zero to one bank and diminishing gains thereafter—the pattern measured in Section 6.

Corollary 1 (Exact abstention)**.** *For every bank and every time, a fingerprint mismatch contributes exactly zero to r_t. Hash collisions can delete or shorten a token's history but cannot create a false positive value under exact integer fingerprints.*

5 Experimental Protocol

All language-model quality results use TinyStories (Eldan & Li, 2023) with a frozen BPE-2048 tokenizer, 2,048,000 training tokens per run, B4×L512, AdamW, and three seeds. Development NLL is evaluated on the same 64 deterministic windows per length at L32/128/512/2048/4096, scoring the final quarter. The primary 11.7M comparison matches model parameters within 1.0%; all nine preregistered runs completed and are included. Earlier 6.1M and 23.7M runs use the same token and evaluation contract. Hierarchical paired bootstrap resamples seeds and aligned windows for 50,000 draws. All reported quality measurements use development splits; no result is presented as held-out test performance.

The TinyShakespeare experiment tokenizes a frozen public text with the same vocabulary and evaluates 32 deterministic windows per length. It was specified after the in-domain scaling analysis and is therefore reported separately as *post-hoc OOD development evidence*; it is not used as a confirmatory held-out result. The bank-count study freezes every learned parameter and evaluates memory disabled and $J \in \{1, 2, 4, 6\}$; no variant is retrained. Full hashes and configurations appear in the supplement.

6 RESULTS

Table 1: Matched 11.7M development NLL (lower is better). RAVEL, Mamba-3, and RWKV-7 use the recovered three-seed contract; GDN2/KDA/CSA are pinned reference-equation baselines under the same data, token, and evaluation contract. This is quality evidence, not official fused-runtime evidence.

Model	L32	L128	L512	L2048	L4096
RAVEL	2.654	2.384	2.479	2.439	2.488
Mamba-3	2.643	2.450	2.556	2.494	2.540
RWKV-7	2.787	2.629	2.743	2.680	2.725
GDN2	2.795	2.613	2.738	2.673	2.717
KDA	3.106	2.922	3.022	2.968	3.022
CSA	2.937	2.832	2.949	2.904	2.958

Matched quality. Table 1 shows the central boundary. At L32, RAVEL and Mamba-3 are statistically indistinguishable under the paired analysis. From L128 through L4096, the explicit memory yields lower NLL. RAVEL also improves over RWKV-7 at every measured length. Pinned GDN2, KDA, and CSA reference-equation baselines are included for quality context only; they are not official fused-runtime comparisons.

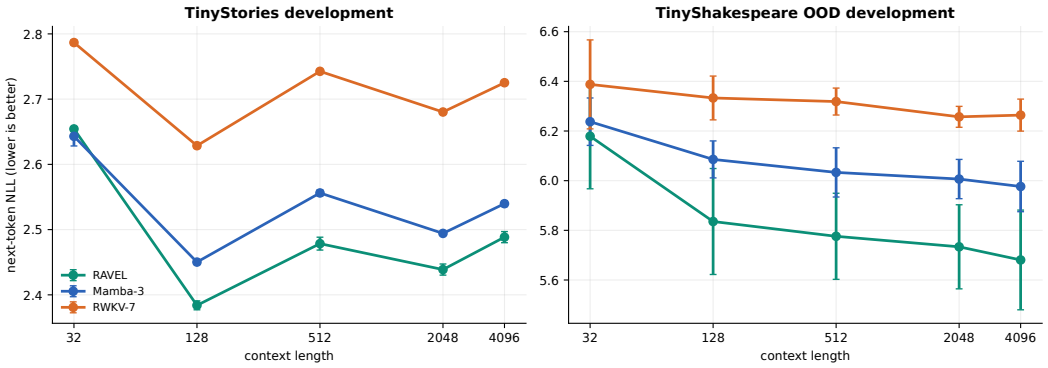


Figure 2: Three-seed quality as context grows. Error bars show seed standard deviation. RAVEL matches the short-context boundary and improves the measured medium/long contexts in-domain (left); the post-hoc OOD development study shows the same qualitative separation (right).

Table 2: RAVEL-minus-baseline paired NLL deltas with 95% hierarchical-bootstrap intervals (50k draws). Negative values favor RAVEL. OOD is explicitly post-hoc development evidence.

Dataset	Baseline	L128	L512	L2048	L4096
TinyStories	Mamba-3	-0.066 [-0.079,-0.054]	-0.078 [-0.089,-0.068]	-0.055 [-0.063,-0.047]	-0.051 [-0.057,-0.046]
TinyStories	RWKV-7	-0.245 [-0.258,-0.231]	-0.264 [-0.278,-0.249]	-0.241 [-0.251,-0.231]	-0.237 [-0.244,-0.229]
Shakespeare OOD	Mamba-3	-0.250 [-0.498,-0.066]	-0.257 [-0.402,-0.146]	-0.273 [-0.413,-0.175]	-0.296 [-0.468,-0.178]
Shakespeare OOD	RWKV-7	-0.498 [-0.698,-0.289]	-0.543 [-0.704,-0.362]	-0.523 [-0.690,-0.350]	-0.583 [-0.770,-0.363]

Scaling. Figure 3 plots paired deltas at 6.1M, 11.7M, and 23.7M parameters. The RAVEL–Mamba difference at L32 remains close to zero; the L128–L4096 advantage persists at all three scales. The gap to RWKV-7 is larger and consistently negative. This is a matched development scaling trend, not a universal power law.

Transfer. The OOD result is stronger at long context (Table 2): RAVEL-minus-Mamba reaches -0.296 NLL at L4096, and RAVEL-minus-RWKV reaches -0.583. Seed variance is large at L32, where both intervals cross zero. Because this benchmark was specified after the in-domain analysis, it provides development evidence of mechanism transfer rather than a confirmatory generalization test.

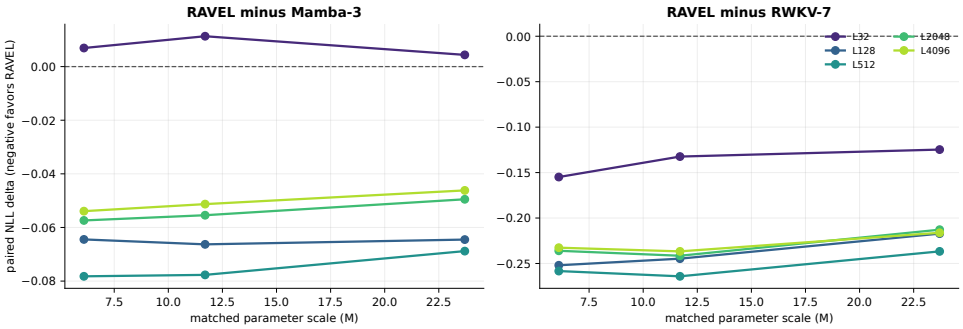


Figure 3: Parameter scaling under a fixed 2.048M-token budget. Each point is the three-seed paired mean. Negative values favor RAVEL. The qualitative short/long boundary persists across the three completed scales.

Table 3: Frozen bank-count ablation. Long NLL averages L128–L4096. J1 recovers most of the memory benefit; extra banks provide collision redundancy with diminishing marginal quality gain.

Variant	In-domain long	OOD long	In-domain L32
No memory	2.5114	5.9140	2.6543
J1	2.4489	5.7660	2.6543
J2	2.4478	5.7589	2.6543
J4	2.4476	5.7567	2.6543
J6 (full)	2.4474	5.7565	2.6543

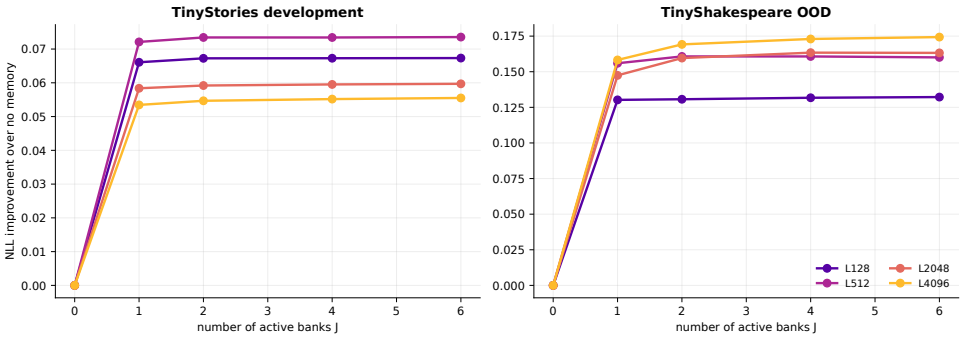


Figure 4: Frozen bank-count ablation. One bank recovers most of the long-context benefit; additional banks produce smaller gains, concentrated at longer contexts where collision exposure is larger. L32 is exactly unchanged because $\rho_t = 0$.

Ablation and mechanism. With memory disabled, the RAVEL checkpoint performs similarly to its recurrent core. Activating a single verified bank improves performance across all evaluated context lengths (L128–L4096). Adding a second bank (J2) yields smaller but statistically significant gains in most long-context settings, whereas J4 and J6 are often statistically indistinguishable except at the longest context lengths. Across bank-count configurations, the valid-read rate correlates with NLL improvement ($r = 0.6907$ in-domain and 0.8616 OOD). These results support the proposed collision-survival mechanism while showing that empirical performance saturates before the theoretical survival probability does.

Table 4: Observed reference-runner resources. These measurements include the auditable Python memory path and are not native prefill/decode claims.

Model	Params	train tok/s	peak alloc.	extra state
RAVEL	11.81M	213.5	10.03 GiB	1.647 MB
Mamba-3	11.64M	270.2	10.02 GiB	–
RWKV-7	11.65M	897.5	1.18 GiB	–

Systems boundary. The present implementation prioritizes an auditable Python reference path. It trains at 213 tokens/s, versus 270 for the matched Mamba path and 898 for RWKV, and allocates about 10.0 GiB. A standalone Triton operator passes output/state/value-gradient checks within 4.77×10^{-7}, but official native full-vs-decode fidelity gates for the baselines did not meet the preregistered tolerance. We therefore report no production prefill/decode superiority. The architecture result is a quality–state trade-off; kernel fusion is future work.

7 Discussion and Limitations

RAVEL's effectiveness stems from a principled division of labor: the recurrent core models distributed dynamics, while a small, verifiable table estimates token-local successor prototypes. Exact fingerprints ensure that collisions manifest as missing reads rather than corrupted ones. The delayed horizon further prevents this inductive bias from imposing unnecessary costs when the context is too short to yield reliable recurrence statistics.

Several limitations are important. First, all primary experiments are conducted on TinyStories with a small BPE vocabulary, so the results do not establish downstream accuracy at scale. Second, the OOD evaluation is post hoc. Third, lexical fingerprints capture associations between repeated tokens rather than semantic nearest-neighbor relationships and therefore cannot replace global attention in arbitrary relational-reasoning tasks. Fourth, the state size remains constant only under a fixed slot budget, and the reference implementation requires 1.65 MB per sequence. Fifth, the fully integrated, auditable path is slower than the baselines. Finally, formal succinctness results show that some languages require recurrent representations exponentially larger than their Transformer counterparts (Bergsträßer et al., 2026); our bounded empirical findings do not contradict this theoretical separation.

8 Conclusion

We introduced RAVEL, a recurrent architecture augmented with collision-verified state. Its causal direct-address pathway estimates conditional successor prototypes and admits an explicit collision-survival law. Experiments across three model scales and three random seeds demonstrate consistent long-context improvements with a sharp short-context boundary. OOD evaluations and frozen-state ablations further identify the verified state pathway as the source of these gains and reveal diminishing returns from increasing bank capacity. Together, this mechanistic evidence and the clearly defined systems boundary constitute our central contribution: a simple, analyzable augmentation that recovers information forgotten by the recurrent state without claiming to address every challenge in sequence modeling.

AI Use Statement

AI tools assisted with code organization, statistical aggregation, figure generation, and drafting. All numerical claims in the paper are generated from frozen result manifests and closure files included in the supplementary package. The authors are responsible for the scientific claims, source verification, and final text.

Reproducibility Statement

The supplementary package contains the canonical architecture source, packaged implementation, semantic tests, frozen configurations, per-seed manifests, preregistrations, closure hashes, aggregation scripts, and complete proofs. Appendix sections document data identities, matched parameter counts, optimizer/evaluation contracts, and the boundary between reference-runner and native systems evidence.

References

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2025.

Pascal Bergsträßer, Ryan Cotterell, and Anthony W. Lin. Transformers are inherently succinct. *arXiv preprint arXiv:2510.19315*, 2026.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, et al. Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv preprint arXiv:2601.07372*, 2026.

Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

Wanyun Cui. A hippocampus for linear attention: An exact memory for what the recurrent state forgets. *arXiv preprint arXiv:2607.02303*, 2026.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*, 2017.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Kimi Team. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. *arXiv preprint arXiv:2603.15569*, 2026.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, et al. Rwkv-7 "goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

RAVEL: COLLISION-VERIFIED STATE AUGMENTATION FOR LONG-CONTEXT RECURRENT LANGUAGE MODELS: SUPPLEMENTARY MATERIAL

Anonymous authors
Paper under double-blind review

A ARCHITECTURE AND OPERATIONAL SEMANTICS

This appendix specifies the implementation used in every reported RAVEL run. The canonical byte-preserved source is included in the code supplement. The packaged module reorganizes imports and configuration but leaves the equations, update order, constants, and tensor shapes unchanged.

A.1 STATE AND NOTATION

For batch element b, bank $j \in [J]$, and slot $s \in [B]$, the memory state is

$$S_{b,j,s} = (f_{b,j,s}, c_{b,j,s}, m_{b,j,s}), \tag{1}$$

where f is an integer token fingerprint, $c \in \{1, \ldots, C\}$ is a capped support count, and $m \in \mathbb{R}^{d_v}$ is a differentiable value. Empty slots have no fingerprint. Address tables $a_j(x)$ are sampled once with seed 773101 and registered as immutable model buffers. The reported configuration uses $V = 2048$, $J = 6$, $B = 1024$, $d_v = 64$, and $C = 4096$.

The recurrent backbone is a four-layer Mamba-3 MIMO reference core with width 608, state dimension 64, and head dimension 64. The RAVEL model adds a value embedding $E_v \in \mathbb{R}^{V \times d_v}$, read projection $R \in \mathbb{R}^{d \times d_v}$, and scalar gate $g : \mathbb{R}^d \to \mathbb{R}$. Its output projection is shared with the recurrent core.

A.2 ONE-STEP ALGORITHM

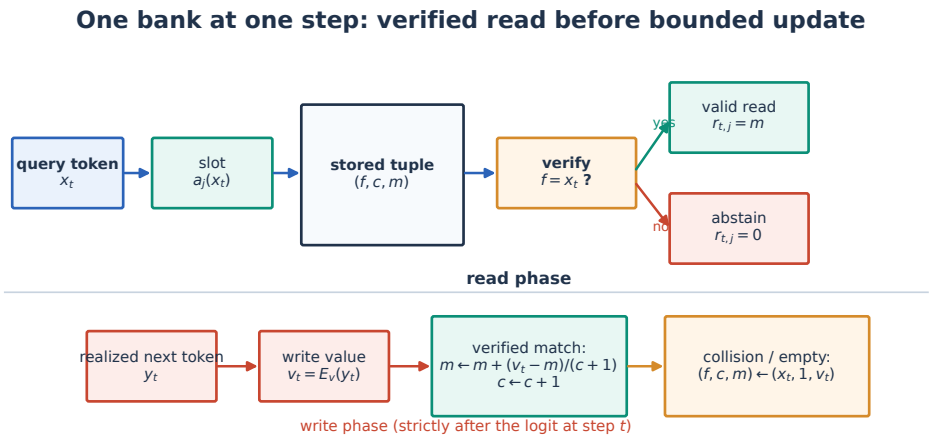


Figure 1: Operational semantics for one bank. Every current read is resolved from the pre-write state. Fingerprint mismatch produces an exact zero. The realized successor value is committed afterward by a convex mean update or a bounded replacement.

For clarity, Algorithm A.2 writes the sequential reference operation. The actual runner vectorizes the neural core but retains this state order.

Algorithm 1: RAVEL step at time t

1. Compute recurrent hidden state $h_t = F_\theta(x_{\le t})$ and base logits $z_t^{\text{core}} = W_o h_t$.
2. For every bank j, address $s_{t,j} = a_j(x_t)$ and read the pre-write tuple $(f, c, m) = S_{t,j,s_{t,j}}$.
3. Set $q_{t,j} = \mathbf{1}\{f = x_t\}$ and $r_{t,j} = q_{t,j} m$. Aggregate $r_t = \sum_j r_{t,j} / \max(1, \sum_j q_{t,j})$.
4. Emit $z_t = z_t^{\text{core}} + \rho_t \sigma(g(h_t)) W_o R r_t$.
5. After y_t is realized, set $v_t = E_v(y_t)$. If $f = x_t$, update $c' = \min(c+1, C)$ and $m' = m + (v_t - m)/c'$. Otherwise replace the tuple by $(x_t, 1, v_t)$.

A.3 Teacher forcing and deployment

The training function accepts tensors $(x_{1:T}, y_{1:T})$ together because the entire loss is teacher-forced. This API does not make y_t available to the current prediction. The state update for (x_t, y_t) occurs after the current read and logit. In incremental generation, the same update is executed after sampling or selecting the realized token and before processing the next step. Therefore the state sequence is the same under a common realized token trajectory.

A.4 Reference storage accounting

The audited reference representation stores an int32 fingerprint, int64 count, and 64 fp32 value coordinates: $4 + 8 + 4 \cdot 64 = 268$ bytes per slot. Hence

$$6 \cdot 1024 \cdot 268 = 1{,}646{,}592 \text{ bytes per sequence.} \tag{2}$$

This is an implementation count, not an information-theoretic minimum. Packing fingerprints/counts or quantizing values can reduce it, but no such optimization is used in the reported results.

B Proofs

B.1 Causality

Theorem 1 (Causality)**.** *At every step t, the RAVEL logit z_t is measurable with respect to the sigma-algebra generated by $(x_{\le t}, y_{<t})$ and fixed model parameters. In particular, z_t is independent of y_t conditional on this history.*

Proof. Proceed by induction. Before step 1 the bank state is fixed and empty. Assume the pre-read state S_t is a deterministic function of $(x_{<t}, y_{<t})$. The recurrent hidden state is a function of $x_{\le t}$. Addresses $a_j(x_t)$ and the equality tests against stored fingerprints use only x_t and S_t, so $q_{t,j}$, $r_{t,j}$, and r_t are functions of $(x_{\le t}, y_{<t})$. Equation (4) in the main paper therefore has the same property. Only after z_t has been formed does the algorithm compute $v_t = E_v(y_t)$ and update the addressed tuples. Consequently S_{t+1} is a function of $(x_{\le t}, y_{\le t})$, establishing the induction hypothesis for the next step. □

Corollary 1 (Teacher-forced/autoregressive state equivalence)**.** *For any fixed realized token sequence, teacher forcing and incremental autoregressive execution produce identical pre-read states and logits at each step, up to the numerical path used by the common neural operators.*

Proof. Both executions start from the same empty state and apply the same post-read update to the same realized pair (x_t, y_t). Induction on t gives equality. □

B.2 Exact abstention

Theorem 2 (No false-positive reads under exact fingerprints)**.** *If slot $a_j(x_t)$ contains fingerprint $f \neq x_t$, then bank j contributes exactly zero to the aggregate at step t.*

Proof. By definition $q_{t,j} = \mathbf{1}\{f = x_t\} = 0$, hence $r_{t,j} = q_{t,j} m = 0$. The implementation applies the integer equality before including the tensor value. The denominator is clamped to one when every bank abstains, so the aggregate remains zero rather than undefined. □

This property separates RAVEL from approximate retrieval: collisions reduce recall but do not substitute another token's prototype. It does not prevent a collided slot from deleting earlier support; deletion is measured by the valid-rate diagnostics and controlled probabilistically by multiple banks.

B.3 Running prototype identity

Lemma 1 (Empirical-mean invariant). *Let $m_1 = v_1$ and $m_n = m_{n-1} + (v_n - m_{n-1})/n$. Then $m_n = n^{-1} \sum_{i=1}^{n} v_i$.*

Proof. The base case is immediate. If $m_{n-1} = (n-1)^{-1} \sum_{i<n} v_i$, then

$$m_n = \left(1 - \frac{1}{n}\right) m_{n-1} + \frac{1}{n} v_n \tag{3}$$

$$= \frac{1}{n} \sum_{i<n} v_i + \frac{1}{n} v_n = \frac{1}{n} \sum_{i=1}^{n} v_i. \tag{4}$$

□

Proposition 1 (Least-squares prototype). *The uncapped state m_n uniquely minimizes $\sum_{i=1}^{n} \|m - v_i\|_2^2$.*

Proof. Differentiate the strictly convex objective: $2nm - 2\sum_i v_i = 0$. The solution is the empirical mean from the preceding lemma. □

Proposition 2 (Convergence). *Suppose verified successor values for token x are i.i.d. with $\mathbb{E}\|E_v(Y)\| < \infty$, no collision interrupts the slot, and the count is uncapped. Then $m_n \to \mathbb{E}[E_v(Y) \mid X = x]$ almost surely.*

Proof. By the empirical-mean invariant, m_n is the sample mean of the conditional successor-value sequence. Apply the vector strong law coordinatewise. □

The deployed count is capped at 4096. Before the cap it is the exact mean. After the cap the update becomes a constant-step exponential smoother with step $1/4096$; it remains bounded but need not converge to a fixed sample mean under nonstationarity.

B.4 Boundedness and non-expansion of the value update

Proposition 3 (Convex-hull boundedness). *If every value embedding satisfies $\|v_t\| \le M$, then every stored value satisfies $\|m\| \le M$.*

Proof. Replacement sets $m = v_t$. A verified update is $(1 - 1/c')m + (1/c')v_t$, a convex combination for $c' \ge 1$. The norm ball of radius M is convex, so induction over updates proves the claim. □

Proposition 4 (Bounded memory correction). *Let operator norms be induced by ℓ_2. Then*

$$\|z_t - z_t^{\text{core}}\| \le \|W_o\| \, \|R\| M. \tag{5}$$

Proof. The mean of valid reads is a convex combination of stored values and has norm at most M. Both ρ_t and $\sigma(g(h_t))$ lie in $[0, 1]$. Apply submultiplicativity. □

This is a bounded-input/bounded-output statement for the memory branch. It does not imply that the entire trained recurrent network is contractive.

B.5 COLLISION SURVIVAL

Theorem 3 (Single-bank survival)**.** *Let K distinct token types be written after the most recent write of x. If a_j is uniform and pairwise independent across token types, the slot for x remains uncollided with probability $(1 - 1/B)^K$.*

Proof. For each interfering type $u \neq x$, $\mathbb{P}[a_j(u) \neq a_j(x)] = 1-1/B$. Under the stated independence, multiply the K probabilities. □

Theorem 4 (Multi-bank survival)**.** *If bank maps are independent, the probability that at least one of J banks retains a verified value is*

$$P_{J,K} = 1 - \left(1 - \left(1 - \frac{1}{B}\right)^K\right)^J. \tag{6}$$

Proof. Let $p_K = (1-1/B)^K$. Each bank fails with probability $1-p_K$. Independence gives all-bank failure probability $(1 - p_K)^J$; take the complement. □

Corollary 2 (Diminishing bank returns)**.** *The incremental survival gain is*

$$P_{J+1,K} - P_{J,K} = p_K(1 - p_K)^J, \tag{7}$$

which decreases geometrically with J for $0 < p_K < 1$.

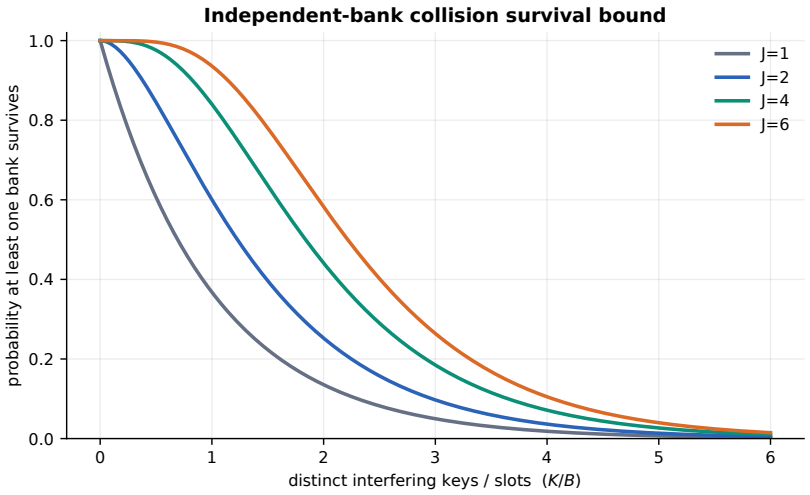


Figure 2: Theoretical probability that at least one independent bank retains a verified value after K distinct interfering keys. The ablation's large J0→J1 gain and diminishing later gains match the predicted shape.

The bound idealizes the frozen random address tables as independent uniform maps and treats distinct interfering types once. Repeated writes, nonuniform lexical frequencies, and correlations in realized sequences change the empirical survival rate; the result is a mechanism model, not an exact language-data law.

B.6 SHORT-CONTEXT IDENTITY

Lemma 2 (L32 identity)**.** *For every sequence of length at most 32, RAVEL and its no-memory ablation produce identical logits.*

Proof. For positions $0 \leq t < 32$, $\rho_t = \mathrm{clip}((t - 64)/32, 0, 1) = 0$. Equation (4) therefore reduces exactly to $z_t = z_t^{\mathrm{core}}$, independent of the memory state. This algebraic identity explains the exactly zero bank-ablation deltas at L32. □

C Additional Results

C.1 Quality curves and paired intervals

The main paper reports means and representative intervals. Machine-readable closures contain every per-window NLL, seed value, interval, and input hash. At 11.7M, RAVEL-minus-Mamba mean deltas at L32/128/512/2048/4096 are $+0.0113, -0.0663, -0.0777, -0.0554, -0.0513$; only L32 crosses zero. RAVEL-minus-RWKV deltas are $-0.1324, -0.2447, -0.2641, -0.2415, -0.2367$, all with intervals below zero.

At 6.1M, RAVEL-minus-Mamba long-context deltas range from -0.0539 to -0.0782; at 23.7M they range from -0.0462 to -0.0688. L32 crosses zero at both scales. RAVEL-minus-RWKV intervals remain below zero for every length at both scales. These results establish persistence across the three completed points; they do not justify extrapolation beyond them.

C.2 Full bank-count ablation

The frozen ablation evaluates memory disabled, J1, J2, J4, and the existing J6 run without changing learned parameters. One bank reduces in-domain mean long-context NLL from 2.5114 to 2.4489 and OOD from 5.9390 to 5.7660. J2, J4, and J6 produce smaller incremental changes. The exact context-wise means and 50k intervals appear in the machine-readable closure.

Valid-read rate and NLL improvement have descriptive Pearson correlations 0.6907 in-domain and 0.8616 OOD over 45 seed–length–bank points. This correlation is compatible with the collision-survival mechanism, but it is not an independent causal estimate: bank count jointly changes the valid rate and number of read replicas.

C.3 Operator evidence

The CPU symbolic/VJP harness covers repeated matches, mismatch/no-write, promotion, support cap, and random forced-collision fixtures. It observes every action type and exactly replays the discrete tape. Initial testing of the production Triton operator revealed a numerical mismatch at power-of-two sequence lengths: compile-time loop specialization altered the floating recurrence while leaving discrete metadata correct. A formula-neutral amendment changed sequence length to a runtime scalar. Table 1 reports the final independent comparison. This validates the memory primitive, not the current full-model systems path.

Table 1: Independent operator validation. Discrete fingerprints/actions replay exactly; floating outputs and gradients are compared to the CPU reference.

Check	Maximum error / result
Read output	4.77×10^{-7}
Final value state	2.38×10^{-7}
Value gradient	4.77×10^{-7}
Initial-state gradient	0
Discrete state and action tape	exact
Reference operator throughput (B4,L512)	3.69M token/s
Peak operator memory	33.97 MB

D Statistical Analysis Details

D.1 Estimands

The primary estimand at scale p, length L, and baseline b is the paired development-window difference

$$\Delta_{p,L,b} = \frac{1}{3W} \sum_{s=1}^{3} \sum_{w=1}^{W} \left(\ell^{\mathrm{RAVEL}}_{p,L,s,w} - \ell^{b}_{p,L,s,w} \right). \tag{8}$$

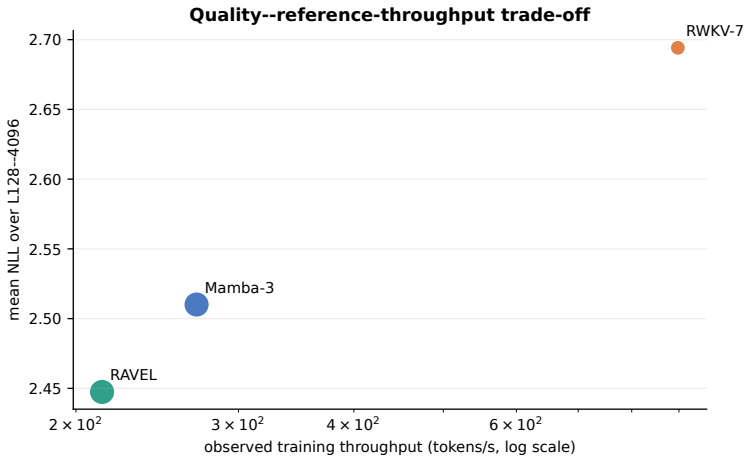


Figure 3: Observed development quality versus reference-runner training throughput. Marker area scales with peak allocated CUDA memory. The figure is deliberately a trade-off, not a systems win: RAVEL improves long-context NLL but is slower and larger in the auditable integration.

Negative values favor RAVEL. The bootstrap samples three seed indices with replacement; for each sampled seed it samples W aligned window indices with replacement. This preserves both seed and window variation while respecting pairing.

No correction for multiple comparisons is applied. We use intervals to characterize a pattern across prespecified lengths, not to declare a family-wise hypothesis test. Exact intervals, not only binary significance labels, are reported.

D.2 Why aligned windows matter

NLL varies substantially across text windows, especially under OOD evaluation. Independent-window resampling would add avoidable content variance. Pairing each architecture on identical token windows estimates the architectural difference conditional on the frozen window population. Seed-level resampling prevents treating 192 correlated windows as 192 independent training runs.

D.3 Ablation comparisons

For a frozen checkpoint, J1–no-memory, J2–J1, J4–J2, and J6–J4 are paired at the seed/window level. L32 differences are exactly zero by the horizon lemma; their bootstrap distribution is degenerate and is labeled inconclusive rather than significant.

E Systems Evidence and Evaluation Boundary

The reference runner records end-to-end training time and peak CUDA allocation. These measurements are useful for disclosing cost but do not compare optimized kernels. RAVEL averages 213.45 token/s and 10.028 GiB peak allocation; matched Mamba averages 270.21 token/s and 10.022 GiB; RWKV averages 897.55 token/s and 1.178 GiB. The memory module therefore does not improve the measured training system.

A preregistered native full-vs-step fidelity gate required maximum functional error below 2×10^{-5} before timing. Available BF16 paths for Mamba-3, HOLA, and RWKV-7 did not satisfy the complete JS/KL/NLL/token-error contract; top-1 agreements were 0.992188, 0.990234, and 0.994141. Under the preregistered protocol, formal native timing was therefore excluded for all evaluated arms; the tolerance was not changed and prototype timing was not substituted.

This systems outcome has two implications. First, the standalone memory operator's 3.69M token/s is evidence that the primitive can be parallelized, not evidence that the present full model is production efficient. Second, the paper's contribution is mechanism and quality under bounded state. A future

systems paper must fuse the memory path and re-establish model-level full/step fidelity before measuring decode.

REFERENCES

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2025.

Pascal Bergsträßer, Ryan Cotterell, and Anthony W. Lin. Transformers are inherently succinct. *arXiv preprint arXiv:2510.19315*, 2026.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, et al. Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv preprint arXiv:2601.07372*, 2026.

Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

Wanyun Cui. A hippocampus for linear attention: An exact memory for what the recurrent state forgets. *arXiv preprint arXiv:2607.02303*, 2026.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*, 2017.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Kimi Team. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. *arXiv preprint arXiv:2603.15569*, 2026.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, et al. Rwkv-7 "goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.