POSITIONS ON TREES, NOT LINES: EXACT LENGTH GENERALIZATION VIA DYADIC ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose dyadic attention, which represents token offsets by their scale and divisibility by powers of two. Its hard-routing version, the Dyadic Scan Transformer (DST), learns finite-monoid composition and computes all prefix products in $\lceil \log_2 n \rceil$ iterations of a single weight-shared block. Trained with intermediate scan supervision on sequences of at most 64 tokens, DST solves parity and the NC^1-complete S_5 word problem with zero token errors on sequences of 1,048,576 tokens: a 16,384-fold length extrapolation in 20 iterations, using codebook projection for S_5. For this projected inference mode with a fixed sentinel, a machine-checked certificate proves correctness at arbitrary lengths while accounting for floating-point error. We also pretrain matched 1.3B- and 3B-parameter dense Llama-style decoder-only LLMs from scratch, adding a learned dyadic bias alongside RoPE without degrading perplexity at the pretraining context length. We derive a closed-form occupancy correction that requires no further training. At 1.3B, extending the dyadic table through finetuning on contexts up to 16k tokens and applying this correction gives PG-19 perplexity of 18.4 at 65,536 tokens (21.6 for long-context-finetuned RoPE), four times the maximum finetuning length. A variant finetuned up to 64k tokens reaches perplexity of 20.9 at 262,144 tokens without correction. A separate finetune combining dyadic table extension with positional interpolation achieves 94% and 75% passkey retrieval accuracy at 8k and 16k, compared with 60% and 22% for our full-budget YaRN-finetuned baseline.

1 INTRODUCTION

A running parity bit is a simple example of state tracking: after each input bit, keep the current parity or flip it. The rule does not change when the sequence gets longer, but a model that fits short examples need not learn a reusable rule. Empirical failures on such tasks are well documented (Anil et al., 2022; Delétang et al., 2023). Expressivity results also identify limitations of fixed-depth transformers under their respective architectural, precision, and complexity assumptions (Hahn, 2020; Merrill et al., 2024). Logarithmic-depth constructions can overcome these limitations (Liu et al., 2023; Merrill & Sabharwal, 2025). Our question is how to make the learned computation reusable across these additional depths, and how to check that it is.

The intuition: reuse one merge rule at every scale. Suppose a position already summarizes its preceding 2^t tokens. Reading the summary 2^t positions to its left lets it combine two adjacent windows into one of width 2^{t+1}. For an associative operation, the same merge rule works at every scale. Doubling the context therefore adds one iteration with the same merge parameters (Fig. 1). This is classical parallel scan (Blelloch, 1990). Our contribution is a transformer realization with a learned local update and a finite, checkable condition for its exact transfer.

We describe an offset by two binary features: its distance scale and its alignment with powers of two. This leads to two distinct uses of dyadic attention. In the **hard-routing** Dyadic Scan Transformer (DST), a structural mask supplies the appropriate scan partner at each iteration. The network learns the composition rule under this fixed routing. In the **soft-bias** language model, a learned table over the same features supplements content attention with rotary positional embeddings (RoPE). Offsets with the same features share a table entry, or cell. As context grows, more keys can share

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.

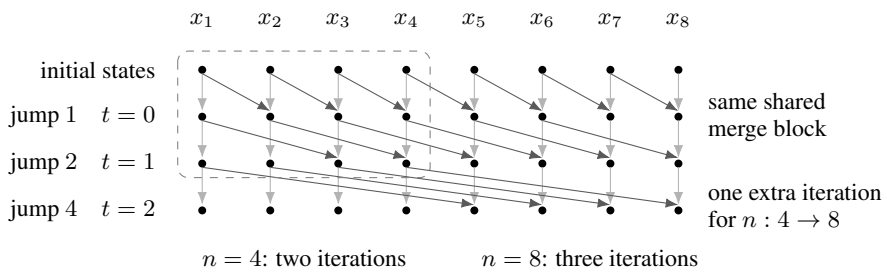

Figure 1: **Doubling context reuses the DST merge block.** Vertical arrows retain the current window summary, and diagonal arrows read its left partner. The dashed region is the length-4 computation. Length 8 repeats the same update and adds jump 4. Boundary positions keep their state through the sentinel, whose edges are omitted.

a cell, motivating an inference-time occupancy correction. This second model uses ordinary causal attention rather than the certified scan.

Contributions and scope. First, we give a shared-block realization of finite-monoid scans and a finite transition criterion for exact generalization to arbitrary lengths (§2–3). Second, models trained on at most 64 tokens reach 1,048,576 tokens in 20 iterations. Projected inference enables a floating-point-aware certificate for a specific trained model (§4). Third, matched language-model pretraining and long-context finetuning separate two questions often conflated in context extension: stable perplexity and accurate retrieval (§5–6). The scan guarantee applies to DST, while the LM results are empirical.

2 Dyadic geometry and the scan model

Appendix A collects the notation, task definitions, and evaluation metrics.

What the two coordinates mean. Number tokens $1, \dots, n$ and reserve position 0 for a beginning-of-sequence (BOS) sentinel. For a positive offset $\Delta = i - j$, define

$$r = v_2(\Delta) = \max\{k \in \mathbb{Z}_{\geq 0} : 2^k \mid \Delta\}, \qquad s = \mathrm{msb}(\Delta) = \lfloor \log_2 \Delta \rfloor. \tag{1}$$

Here s is the highest set-bit index, so $2^s \leq \Delta < 2^{s+1}$. The lowest set-bit index r counts trailing binary zeros. For example, $12 = (1100)_2$ has $(r, s) = (2, 3)$, whereas $8 = (1000)_2$ has $(3, 3)$. Thus r measures divisibility by powers of two. An offset is a power of two exactly when $r = s$.

At scan iteration t, we use shifted features $(r - t, s - t)$, optionally clipped coordinatewise to $[-W, W]$, where the integer $W \geq 1$ bounds the feature range. The pair $(0, 0)$ selects exactly $\Delta = 2^t$. Multiplying an offset by two and advancing one iteration preserves the shifted features (Lemma 1), independently of learned weights.

The computation to be learned. A finite monoid $(M, \circ, 1_M)$ is a finite state set with an associative binary operation and an identity: $(a \circ b) \circ c = a \circ (b \circ c)$ and $1_M \circ a = a \circ 1_M = a$. Commutativity is not required. Given $x_1, \dots, x_n \in M$, a prefix scan outputs $y_i = x_1 \circ \cdots \circ x_i$ at every position. Before iteration t, the intended state is the ordered product over the preceding window of width 2^t:

$$w_i^t = \bigcirc_{j=\max(1, i-2^t+1)}^{i} x_j, \qquad w_i^{t+1} = \begin{cases} w_{i-2^t}^t \circ w_i^t, & i > 2^t, \\ w_i^t, & i \leq 2^t. \end{cases} \tag{2}$$

The left-to-right order matters when composing permutations in S_5, a noncommutative example.

Architecture: fixed routing, learned update. DST repeats one pre-LayerNorm transformer block (attention and a multilayer perceptron, MLP), followed by a shared loop LayerNorm. The attention support, or set of readable key positions, is

$$S_t(i) = \{i - 2^t \text{ if } i > 2^t,\ i,\ 0\}. \tag{3}$$

Each head has three learned scalar logits for partner, self, and BOS, with no query–key content score. When the partner is missing, attention is renormalized over self and BOS. There are consequently two length-independent mixing patterns: combine and boundary/keep. Embeddings, mixing logits, value/output maps, MLP, normalizations, and readout are learned. Offsets, the mask, and the iteration count are fixed. The loop index enters only through the mask. A gather-and-mix implementation takes $O(n)$ work per iteration and $O(n \log n)$ total work, with $\lceil \log_2 n \rceil$ parallel iterations. Figure 4 and Appendix D.1 show the shared block, its recurrence, and the masked-attention equations.

Training and projection. We train on freshly generated sequences with $n \sim U[2, 64]$, supervising both prefix outputs and the intermediate window states in Eq. (2). Loop jitter adds extra iterations. Small state noise encourages robustness around the learned states. For tasks where raw recurrent states drift, projection inference maps each post-loop state to its nearest training-derived class centroid in Euclidean distance. These centroids form a finite codebook. We fix the tie-breaking rule. Positive certified margins exclude ties. Centroids are collected from short sequences using task-computable window labels, without long-sequence supervision. The fixed-sentinel certificate additionally holds the sentinel at a training-derived constant. Training and deployment variants are distinguished in §4. Full recipes are in Appendix D.

3 What can be guaranteed

We distinguish representability from a guarantee for a particular learned model. Appendix B gives proofs and obstruction results.

Lemma 1 (Dilation shift). *For $\Delta > 0$, $v_2(2\Delta) = v_2(\Delta) + 1$ and $\mathrm{msb}(2\Delta) = \mathrm{msb}(\Delta) + 1$. Hence the shifted, clipped features agree at (Δ, t) and $(2\Delta, t+1)$, and $S_{t+1}(2i) = 2S_t(i)$.*

Theorem 1 (Uniform log-depth scan). *For every finite monoid M, a sufficiently wide* DST *with a two-layer ReLU MLP admits one parameterization, independent of n, that computes all prefix products after $\lceil \log_2 n \rceil$ iterations in exact arithmetic. Both hard selection and a suitable choice of positive, fixed softmax mixtures on the three-role support admit such a construction. The sentinel can be fixed.*

The mask specifies communication. The constructed update maps finitely many state/read patterns to the correct combine or keep state. It also realizes transition-monoid scans for regular languages. The S_5 word problem is NC^1-complete under AC^0 reductions (Barrington, 1989). Our model computes the group scan; it does not implement the reductions from other NC^1 problems. The representability result is separate from whether gradient descent finds the construction or whether a floating-point implementation is exact.

Theorem 2 (Exact transfer from a finite transition table). *Let $e : M \to \mathbb{R}^d$ be an injective codebook of d-dimensional state vectors with a correct output decoder. Initialize token i to $e(x_i)$ and fix the sentinel to f. Write Φ_{comb} and Φ_{keep} for the two effective shared-block updates, including their fixed attention mixtures and sentinel contribution. In $\Phi_{\mathrm{comb}}(x, y; f)$ the first argument is the query position's own state and the second is the state read from its partner at offset 2^t; this convention fixes the composition order $b \circ a$ below. If, for all $a, b \in M$,*

$$\Phi_{\mathrm{comb}}(e(a), e(b); f) = e(b \circ a), \qquad \Phi_{\mathrm{keep}}(e(a); f) = e(a), \tag{4}$$

then every n and every $T' \geq \lceil \log_2 n \rceil$ produce the correct prefix products at every position.

Corollary 1 (Projected exact transfer). *Replace each effective update by nearest-centroid projection P after that update. If every combine and keep output projects to the required codeword, Eq. (4) holds exactly for the projected model. For learned input embeddings outside the codebook, one must also verify the finite base case; the length-one, zero-loop case requires the output decoder to be correct on the raw input embeddings themselves. Floating-point deployment additionally requires an error enclosure that keeps each audited output inside its correct projection region.*

Projection returns each token to a finite set of canonical states. Consequently the next iteration encounters an already checked local case, rather than a new state with accumulated numerical error.

Why the sentinel, and why the mask? The sentinel separates "there is no partner" from "the partner happens to hold my state." A normalized, state-only attention read cannot distinguish these

Table 1: **Raw recurrence versus projected recurrence.** The 65,536-token columns are exact-sequence accuracy. The final column is token accuracy in the 1,048,576-token evaluation (about 3.1×10^6 tokens). Raw inference uses the learned continuous states without projection; projected inference maps each non-sentinel post-loop state to its nearest fixed codeword (§2).

	Exact-sequence accuracy at 65,536		Token accuracy at 1,048,576
Task	Raw	Projected	Value (inference mode)
Parity	1.00	1.00	1.0000000 (raw)
Carry	1.00	1.00	1.0000000 (raw)
$\mathbb{Z}_{60}$	0.00	1.00	1.0000000 (projected)
S_5	0.00	1.00	1.0000000 (projected)

cases when all values coincide. Proposition 2 makes this obstruction precise for a canonical per-loop scan interface. Its assumptions concern that interface, rather than all sentinel-free transformers. Likewise, finite logits of bounded range cannot concentrate on one key among arbitrarily many competitors (Proposition 3). DST removes those competitors structurally. A soft dyadic LM does not inherit this guarantee simply by using a finite positional table.

4 Length Generalization in State Tracking

Evaluation protocol. We test length extrapolation, use ablations to distinguish routing failures from state drift, and verify when finite transition checks imply correctness beyond the tested lengths. All tasks use prefix scans. Parity and $\mathbb{Z}_{60}$ accumulate sums modulo 2 and 60, respectively. The S_5 task composes permutations of five elements (120 states). The three-state carry monoid composes kill/propagate/generate maps for binary addition, with a task-specific readout. Definitions and metrics are collected in Appendix A.

The main evaluation uses 32 sequences per length and three seeds where specified. Several baseline entries use seed 0 only. Token accuracy is the fraction of correct prefix labels. Exact-sequence accuracy is the fraction of sequences with every label correct. Tables 5–6 (Appendix C) compare all baselines through 65,536 tokens. Selected runs extend to 262,144. Table 1 separately reports the 1,048,576-token tests for parity, carry, S_5, and $\mathbb{Z}_{60}$.

Extrapolation and baselines. Figure 2 shows the parity length sweep. Raw DST is exact on parity and carry through 65,536 for all reported seeds. Parity and carry reach 2^{20} in 20 iterations with zero token errors under raw inference. For carry, both raw and projected inference are error-free on the same three sequences (3,145,728 tokens per mode). On the single-GPU setup in Appendix D, parity takes about 2 seconds per sequence. Projected S_5 and $\mathbb{Z}_{60}$ also reach 2^{20} with zero token errors, at 2.7 seconds per sequence. The long short-term memory network (LSTM) solves parity but requires 43.7 seconds at 2^{20}. Its S_5 sequence accuracy is 0.94 at 65,536. Fixed-depth transformer baselines fail at much shorter lengths. ALiBi is strong on random carry inputs, but its accuracy drops to 0.005 on a retained-carry test at 4096, versus 1.0000 for DST (Appendix I). A separate prompted-LLM comparison is reported in Appendix L.

Routing and state stability. The tested free-table models fit short examples and then lose accuracy under length extrapolation. Appendix J specifies their numerical indexing. Hard-routing DST achieves the reported extrapolation under its training recipe. A separate failure remains: for S_5 and $\mathbb{Z}_{60}$, small state errors accumulate even with correct routing. Projection removes this drift in the reported evaluations. RoPE and looped-NoPE still fail with matched scan supervision. A log-depth recurrent unit (LDRU)-style gather control also exhibits raw state drift. Projection restores its $\mathbb{Z}_{60}$ accuracy through 65,536 (Appendix I). Routing and supervision controls are reported in Appendices G–H. Learned-routing probes and codebook visualizations appear in Appendices J–K.

Accuracy and certification. The fixed-sentinel model with codebook projection passes a floating-point-aware finite verification: for S_5, 14,520 base configurations and 14,520 transitions. The min-

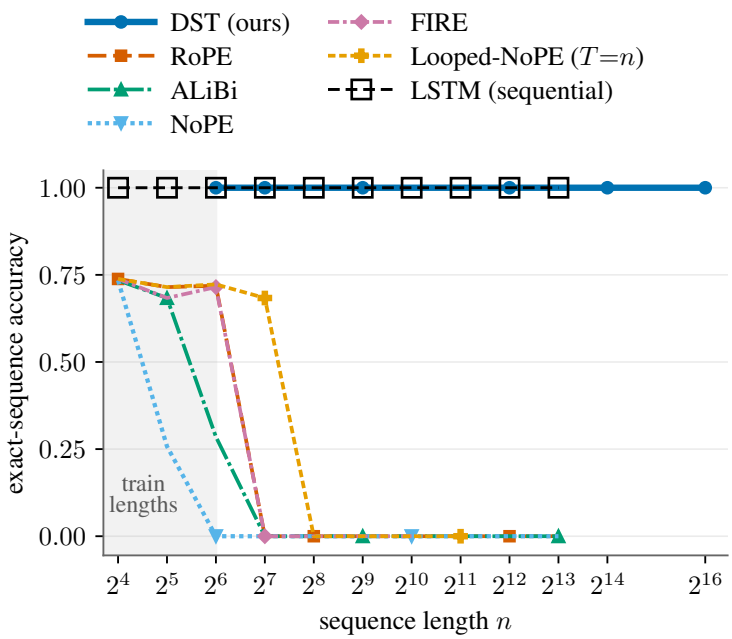

Figure 2: Parity exact-sequence accuracy versus evaluation length, trained at $n \leq 64$ (shaded band). The million-token extension is described in §4. The LSTM is a sequential baseline rather than a logarithmic-depth parallel control. Where curves coincide exactly (accuracy 1.0 or 0.0), they are drawn at their true values and distinguished by marker shape and fill.

imum certified margins are 9.83 for the base case and 10.54 for the transition table. The maximum componentwise output radius is 1.1×10^{-3}. For parity the minimum margin is 4.42 and the radius is 2×10^{-5}. The certificate uses IEEE-754 bounds and a rational enclosure of the error function erf. Under the verified arithmetic and initialization assumptions, induction gives arbitrary-length correctness for that deployment. Sentinel projection also preserves empirical accuracy at 4096 and 65,536. A separate float64 numerical audit enumerates the live-sentinel transitions required through 2^{40} without the same rounding-error enclosures. Appendices E–F distinguish these checks and give the remaining limitations of raw-inference verification.

5 From Hard Routing to a Soft Positional Bias

The same geometry in a language model. For the LMs, we retain ordinary causal content attention. Each head and layer learns an additive table b_c indexed by

$$c(\Delta) = (\min(v_2(\Delta), W), \min(\mathrm{msb}(\Delta), W')), \qquad \ell_{ij} = s_{ij}^{\mathrm{RoPE}} + b_{c(i-j)}, \quad j < i. \tag{5}$$

Here $s_{ij}^{\mathrm{RoPE}} = \widetilde{q}_i^{\top} \widetilde{k}_j / \sqrt{d_h}$ is the content score of the RoPE-rotated query and key, d_h is the head width, and the nonnegative integers W, W' cap the two coordinates. Unlike the scan features, these coordinates are unshifted: there is no scale-indexed loop or hard scan mask. The offset-to-cell map is fixed. The table and the rest of the LM are learned. Clipping gives a finite parameter set, but does not guarantee that all cells were populated in training or that their occupancies remain unchanged. Equation (5) and the analysis below concern positive offsets. The LM correction excludes self (offset zero). BOS is processed as an ordinary causal token.

Why an occupancy correction? If a cell contains twice as many otherwise comparable keys, its total unnormalized softmax weight doubles even though its learned per-key bias has not changed. Subtracting $\log 2$ from that bias compensates for this count change. For a horizon H, define

$$K_H(c) = |\{\Delta : 1 \leq \Delta < H,\ c(\Delta) = c\}|, \quad b_c^{(H)} = b_c - \lambda \log \frac{K_H(c)}{K_T(c)}. \tag{6}$$

Here T is the reference horizon and λ controls the correction strength. The ideal ratio requires $K_T(c), K_H(c) > 0$. The following statement specifies what the counting argument establishes on this support.

Proposition 1 (Occupancy correction on common support)**.** *Let $\mathcal{C}_+$ be the set of cells populated at both horizons T and H. Assume the exponential moment $\mu_c = \mathbb{E}[e^{s_\Delta}]$ of the content score is finite, positive, and shared across offsets Δ in cell c and across these horizons. For additive cellwise corrections $\delta_H(c)$, define $Z_H(c) = e^{b_c+\delta_H(c)} \sum_{\Delta\in c,\, \Delta<H} e^{s_\Delta}$. With $\delta_T = 0$, preserving all ratios $\mathbb{E}Z_H(c)/\mathbb{E}Z_H(d)$ from horizon T for $c, d \in \mathcal{C}_+$ is equivalent to*

$$\delta_H(c) = -\log \frac{K_H(c)}{K_T(c)} + a_H, \tag{7}$$

where a_H is common to those cells. Thus Eq. (6) with $\lambda = 1$ is unique up to a common additive constant within this class.

The result preserves ratios of expected unnormalized masses on common positive support. Expected normalized attention $\mathbb{E}[Z_H(c)/\sum_d Z_H(d)]$ has a random denominator and requires a separate analysis. Cells with $K_T(c) = 0$ also lie outside the result. Uniqueness holds only within the stated correction class (Appendix B). LM accuracy is evaluated empirically.

In the experiments, we apply the count floor to both horizons, $\widetilde{K}(c) = \max\{1, K(c)\}$ for K_T and K_H in Eq. (6), and fix $T = 2048$, including after long-context finetuning. The reference floor extends the correction heuristically to cells empty at T. For a table extended through $\mathrm{msb} = 15$, the nonzero correction at 65,536 tokens acts only on these cells. Its benefit there is empirical, outside the common-support proposition. Appendix P.1 tests the zero-reference policy.

We use full-window counts. Appendix P compares them with causal per-query counts and gives the implementation conventions. Appendix Q examines sparse reads and other consequences of this geometry.

6 Language Modeling and Retrieval

Evaluation design. Matched pretraining tests in-distribution quality. We then vary table depth, finetuning, and occupancy correction to assess perplexity extrapolation, and test retrieval separately. Additional controls examine seeds, model scale, and inference efficiency.

Models and training. The main pair consists of dense 1.3B Llama-style decoder-only models (24 layers, width 2048, 32 heads), trained from scratch for 20,000 updates on about 21B FineWeb-Edu tokens (Penedo et al., 2024) at context 2048. Data order, seeds, and schedules match. The dyadic model adds a zero-initialized 9×9 table per head per layer ($+0.005\%$ parameters), called the shallow table below. The pair uses four H100s per model. We compare evaluation-only corrections with no gradient updates, finetuned configurations, and a reduced-batch comparison of YaRN and positional interpolation (PI).

The deep finetuning configuration extends the dyadic table's scale axis from $\mathrm{msb} = 8$ to $\mathrm{msb} = 15$. We finetune all parameters for 1500 updates at global batch 512×2048, with every twelfth update using a 16,384-token context. The trainer holds tokens per update constant: 7.5% of pretraining updates and tokens (1.57B of 21B). Estimated total dense-model FLOPs are about 8% of pretraining (attention-only work: $\approx 12\%$). This is an analytical estimate (Appendix O). The joint dyadic+PI configuration uses this same 1500-update, batch-512 recipe with interpolation scale $1/16$. YaRN-FT matches this token budget and context schedule exactly (1,572,864,000 tokens in both). Its learning rate, optimizer, and data shard differ. YaRN uses NTK-by-parts interpolation and attention temperature scaling (Peng et al., 2023). Appendix N specifies the settings. Full configurations are in Appendices N, P, and O. Below, FT denotes finetuning, deep-16k/deep-64k name the maximum finetuning context. PI with factor κ rescales rotary position indices by $1/\kappa$.

Perplexity: what is scored. Perplexity is the exponential of the mean token negative log-likelihood. The PG-19 protocol (Rae et al., 2019) scores the second half of each non-overlapping window. Because varying window length also changes scored tokens, a second protocol fixes each

Table 2: **Selected PG-19 results at 1.3B.** Second-half-window perplexity; lower is better. "Occ." denotes occupancy correction; "raw" omits this correction. All pretrains use context 2048. Appendix M gives the complete PG-19 grid, validation results, and fixed-target evaluation. The raw deep-64k checkpoint also reaches 20.9 at 262,144 tokens.

Configuration	2048	32,768	65,536
RoPE, pretrained	17.8	480.4	444
RoPE + eval-time YaRN	17.8	25.9	34.6
RoPE (16k FT)	17.7	19.9	21.6
Deep dyadic (16k FT), raw	17.4	19.5	21.7
Deep dyadic (16k FT) + occ.	17.4	18.1	18.4
Deep dyadic (64k FT), raw	17.8	19.0	19.5
YaRN (16k FT, full budget)	17.9	17.6	18.5

Table 3: **Passkey retrieval at 8k and 16k.** Accuracy with 95% Wilson intervals, 125 trials per length. All four rows use 16k-context finetuning from 2k pretrains. Joint dyadic+PI and deep+occ are separate checkpoints. All rows are 0/125 at 32k (upper Wilson limit 0.03).

Configuration	8192	16,384
PI($\kappa = 16$) FT	.77 [.69, .83]	.55 [.47, .64]
Deep dyadic FT + occ.	.18 [.13, .26]	.03 [.01, .08]
Joint dyadic+PI FT	**.94** [.88, .97]	**.75** [.67, .82]
YaRN-FT (full token budget)	.60 [.51, .68]	.22 [.16, .30]

final 2048-token target block and varies only its preceding context. Its shortest-context result provides a local-context reference for the same targets.

Pretraining, extension, and a strong baseline. Table 2 compares long-context perplexities, and Fig. 3 shows their dependence on context and training step. At context 2048, held-out FineWeb-Edu perplexities are 9.33 versus 9.34 under the uniform local protocol (9.91 versus 9.94 in training-run validation). At 4096, dyadic improves PG-19 perplexity from 68.3 to 62.0, but the uncorrected shallow model is worse than RoPE at some longer lengths. Correction improves that shallow model, yet remains much worse than evaluation-only YaRN: 51.8 versus 18.2 at 4096. The deep finetune gives the strongest measured dyadic extrapolation.

Deep-16k with correction reaches 18.4 at 65,536, versus 21.6 for the long-context RoPE finetune. Deep-64k reaches 20.9 without correction at 262,144. YaRN-FT has lower perplexity at 32k (17.6), reaches 18.5 at 65k, and 21.7 at 131k. Its 262k evaluation exceeded local GPU memory. On fixed target tokens, deep-16k+occ is 18.3 at 65k versus its own 18.5 local reference. YaRN-FT is stronger at 4k–32k but reaches 19.3 at 65k versus its own 18.6 reference (Appendix M).

Passkey retrieval. The passkey task inserts a five-digit key at five depths in PG-19 filler, then asks for it under greedy decoding. The main comparison uses 25 trials per depth, or 125 per length. Wilson 95% intervals describe prompt sampling uncertainty conditional on a fixed checkpoint. Appendix M reports all seven configurations under this 125-trial protocol. Appendix U gives the broader retrieval tests.

In Table 3, the joint dyadic+PI finetune has the highest 8k/16k passkey accuracy among these configurations, whereas deep+occ has stable perplexity but weak retrieval. Applying the correction on top of the joint model hurts both metrics in the reported tests. PI changes the rotary content pathway while leaving the dyadic cell map fixed. This interaction lies beyond Proposition 1, which requires stationary content moments.

Retrieval beyond the finetuning range. All 16k-range headline configurations fail at 32k. Intermediate lengths show dependence on key-to-question distance, followed by more general degrada-

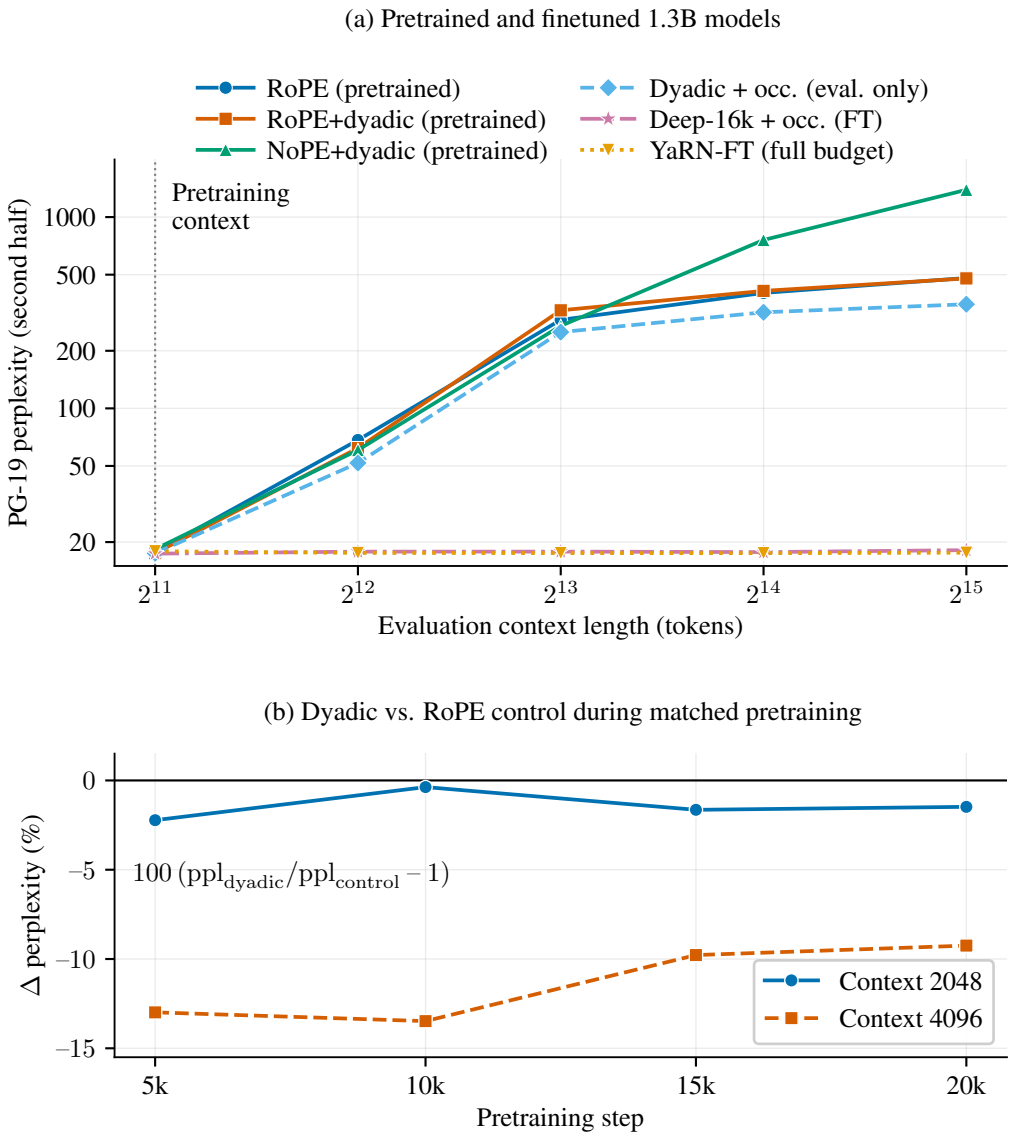

Figure 3: **1.3B PG-19 length generalization.** **(a)** Second-half-window perplexity of the final pretraining checkpoints (RoPE, RoPE+dyadic, NoPE+dyadic; 20k steps, 21B tokens, context 2048), plus evaluation-only occupancy correction, a 1500-step deep-16k finetune with correction, and full-budget YaRN-FT. The two finetunes use matched token budgets. **(b)** Relative perplexity of the matched RoPE+dyadic and RoPE pretrains at the same training step, $100\,(\mathrm{ppl}_{\mathrm{dyadic}}/\mathrm{ppl}_{\mathrm{control}} - 1)$, at evaluation contexts 2048 and 4096. Negative values favor dyadic.

tion. A joint $\kappa = 32$, 32k-context finetune raises 32k passkey accuracy to .27 [.20, .36] and accuracy on the repeated-noise task S-NIAH-2 of the RULER retrieval benchmark to .72. Perplexity stays 18.8–18.9 through 32k, but retrieval fails at 65k and short-context, fine-grained retrieval degrades. This suggests a range–precision tradeoff in this comparison, where finetuning range, interpolation factor, and training budget change together. RULER distractors, multiple keys, and multi-hop variable tracking reveal further failures (Appendices T and U; Hsieh et al., 2024).

Controls. An msb-only table benefits strongly from correction, while a v_2-only table has nearly unchanged perplexity: its approximately stationary relative occupancies make most of the correction

a common softmax shift. The joint two-axis finetune has higher retrieval than the msb-only control, which uses $1/15$ of its token budget and freezes embeddings/readout. These differences prevent isolating the second axis's effect. The 19M/150M controls include ALiBi and NoPE (Appendix R). Appendix S reports the additional pretraining and finetuning seeds. Tables 13–14 report full-model prefill throughput.

Scaling to 3B. We pretrain a matched 3.04B pair on 40B FineWeb-Edu tokens at context 2048. Both models have PG-19 perplexity 15.1 at 2048 (rounded). At 65,536, RoPE, raw dyadic, and corrected dyadic score 499.4, 468.9, and 224.6, respectively. This single-seed test uses pretraining and inference-time correction only (Appendix S, Table 19, Fig. 12).

7 Related work

Learned scans and looped computation. Parallel scan is classical (Blelloch, 1990; Martin & Cundy, 2018). Universal and looped transformers (Dehghani et al., 2019; Giannou et al., 2023; Fan et al., 2025) and log-depth expressivity results (Liu et al., 2023; Merrill & Sabharwal, 2025) provide the computational context. RegularGPT (Chi et al., 2023) already combines weight sharing, logarithmic depth, and sliding-dilated attention. At chunk size two, it reads self and a partner at offset 2^t. Our focus is finite-state closure and floating-point-aware certification of a trained projected scan. WAT (Berezkin, 2026) and an LDRU-style gather model (Pert et al., 2026) provide empirical controls. Appendix V details the architectural distinctions.

Position and context extension. RoPE, ALiBi, NoPE, and functional or hierarchical encodings (Su et al., 2021; Press et al., 2022; Kazemnejad et al., 2023; Li et al., 2024; He et al., 2024; Golovneva et al., 2024) address position through different inductive biases. PI and YaRN (Chen et al., 2023; Peng et al., 2023), LongRoPE/2 (Ding et al., 2024; Shang et al., 2025), TAPA (Wang et al., 2025), and DroPE (Gelberg et al., 2025) modify or remove rotary position dependence. Our occupancy analysis concerns a different, restricted class: additive cellwise corrections under explicit moment and support assumptions.

Multiscale attention. H-Transformer-1D, FMA, LongNet, PowerAttention, and SCOPE (Zhu & Soricut, 2021; Kang et al., 2023; Ding et al., 2023; Chen et al., 2025; Qi et al., 2026) use hierarchical or exponentially spaced structures. SCOPE is particularly relevant because it obtains position information from per-head look-back scopes. Related work studies emergent wavelet-like attention (Ruscio et al., 2024) and explicit wavelet encodings (Irani & Metsis, 2025). The distinction here is the binary scale/divisibility parameterization and the separate analyses of scan transfer and cell occupancy. Appendix V discusses these connections further.

8 Limitations and conclusion

The hard scan model learns state updates with a supplied routing topology and intermediate supervision. Its arbitrary-length certificate concerns a particular fixed-sentinel, projected model with a verified base case and specified arithmetic assumptions. Observed raw accuracy and the soft LM results have no such guarantee. The occupancy theorem controls first moments on common positive support, rather than normalized attention in general. Newly occupied cells require an implementation convention outside that theorem.

LM results separate perplexity calibration from retrieval. Multiple-key and deep variable-tracking probes remain difficult. The 3B pair has one seed, plain ALiBi/NoPE controls stop at 150M, and direct checkpoint comparisons with LongRoPE2, TAPA, or DroPE remain untested. Finetuning settings vary across comparisons, although the main dyadic and YaRN finetunes use the same token budget. On an H100, full-model dyadic prefill takes 7–41% more time per token than the SDPA control. Backward and decoding are not benchmarked (Appendix O). Within these limits, dyadic geometry supplies a reusable communication rule for exact scans and a tractable positional prior for empirical context extension.

Reproducibility statement

Appendix B gives the mathematical arguments and their assumptions. Appendices D and O document the training and evaluation configurations. The supplementary materials provide checker code and inputs, exported codebooks and sentinels, recorded outputs, selected LM implementation and evaluation files, and the scripts and input data used to generate the figures. Appendix W describes these artifacts and the associated evaluation protocols. The arithmetic and deployment scope of the certificate is stated in Appendix E.

AI use statement

Large language models were used under human direction to implement training and evaluation code, assist with experiment execution and monitoring, and revise the manuscript for clarity and consistency. The research ideas and method design were developed by the authors. The authors take responsibility for the final text, mathematical claims, code, and results.

Ethics statement

The experiments use synthetic sequence tasks and existing text corpora; they do not involve participant recruitment or interventions. The state-tracking certificates concern a specified computation, not the safety, privacy, or factual reliability of a language model. The reported perplexity and retrieval metrics do not evaluate those downstream risks. Any redistribution of data or pretrained components must respect their licenses.

References

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2207.04901.

David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989. doi: 10.1016/0022-0000(89)90037-8. URL https://www.sciencedirect.com/science/article/pii/0022000089900378.

Igor Berezkin. Wave-Attractor-Tree: A hierarchical binary tree reduction architecture for efficient sequence modeling. *arXiv preprint arXiv:2603.00812*, 2026. URL https://arxiv.org/abs/2603.00812.

Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990. URL https://www.cs.cmu.edu/~scandal/papers/CMU-CS-90-190.html.

Lida Chen, Dong Xu, Chenxin An, Xintao Wang, Yikai Zhang, Jiangjie Chen, Zujie Liang, Feng Wei, Jiaqing Liang, Yanghua Xiao, and Wei Wang. PowerAttention: Exponentially scaling of receptive fields for effective sparse attention. *arXiv preprint arXiv:2503.03588*, 2025. URL https://arxiv.org/abs/2503.03588.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023. URL https://arxiv.org/abs/2306.15595.

Ta-Chung Chi, Ting-Han Fan, Alexander Rudnicky, and Peter Ramadge. Transformer working memory enables regular language reasoning and natural language length extrapolation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5972–5984. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.397. URL https://aclanthology.org/2023.findings-emnlp.397/.

Hanseul Cho, Jaeyoung Cha, Pranjal Awasthi, Srinadh Bhojanapalli, Anupam Gupta, and Chulhee Yun. Position coupling: Improving length generalization of arithmetic transformers using task structure. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.20671.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2307.08691.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1807.03819.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural networks and the Chomsky hierarchy. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2207.02098.

Chunyuan Deng, Yizhe Zhang, Rui-Jie Zhu, Yuanyuan Xu, Jiarui Liu, T. S. Eugene Ng, and Hanjie Chen. LT2: Linear-time looped transformers. *arXiv preprint arXiv:2605.20670*, 2026. URL https://arxiv.org/abs/2605.20670.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. LongNet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*, 2023. URL https://arxiv.org/abs/2307.02486.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*, 2024. URL https://arxiv.org/abs/2402.13753.

Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent English? *arXiv preprint arXiv:2305.07759*, 2023. URL https://arxiv.org/abs/2305.07759.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2409.15647.

Yoav Gelberg, Koshi Eguchi, Takuya Akiba, and Edoardo Cetin. Extending the context of pretrained LLMs by dropping their positional embeddings. *arXiv preprint arXiv:2512.12167*, 2025. URL https://arxiv.org/abs/2512.12167.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Olga Golovneva, Tianlu Wang, Jason Weston, and Sainbayar Sukhbaatar. Contextual position encoding: Learning to count what's important. *arXiv preprint arXiv:2405.18719*, 2024. URL https://arxiv.org/abs/2405.18719.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020. doi: 10.1162/tacl_a_00306. URL https://arxiv.org/abs/1906.06755.

Zhenyu He, Guhao Feng, Shengjie Luo, Kai Yang, Liwei Wang, Jingjing Xu, Zhi Zhang, Hongxia Yang, and Di He. Two stones hit one bird: Bilevel positional encoding for better length extrapolation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17858–17876. PMLR, 2024. URL https://proceedings.mlr.press/v235/he24c.html.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024. URL https://arxiv.org/abs/2404.06654.

Habib Irani and Vangelis Metsis. DyWPE: Signal-aware dynamic wavelet positional encoding for time series transformers. *arXiv preprint arXiv:2509.14640*, 2025. URL https://arxiv.org/abs/2509.14640.

Yanming Kang, Giang Tran, and Hans De Sterck. Fast multipole attention: A scalable multilevel attention mechanism for text and images. *arXiv preprint arXiv:2310.11960*, 2023. URL https://arxiv.org/abs/2310.11960.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2305.19466.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. Functional interpolation for relative positions improves long context transformers. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.04418.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.10749.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1709.04057.

André F. T. Martins and Ramón F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1614–1623. PMLR, 2016. URL https://proceedings.mlr.press/v48/martins16.html.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. Transformers can do arithmetic with the right embeddings. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.17399.

William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. *arXiv preprint arXiv:2503.03961*, 2025. URL https://arxiv.org/abs/2503.03961.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35492–35506. PMLR, 2024. URL https://proceedings.mlr.press/v235/merrill24a.html.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024. URL https://arxiv.org/abs/2406.17557.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023. URL https://arxiv.org/abs/2309.00071.

Charles Pert, Dalal Alrajeh, and Alessandra Russo. Length generalization with log-depth recurrent units. *arXiv preprint arXiv:2605.26035*, 2026. URL https://arxiv.org/abs/2605.26035.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2108.12409.

Qingguo Qi, Hongyang Chen, and Zhao Li. SCOPE: Boosting LLM efficiency with scoped position encoding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35653–35673. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1650. URL https://aclanthology.org/2026.acl-long.1650/.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019. URL https://arxiv.org/abs/1911.05507.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. Randomized positional encodings boost length generalization of transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1889–1903. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-short.161. URL https://aclanthology.org/2023.acl-short.161/.

Valeria Ruscio, Umberto Nanni, and Fabrizio Silvestri. Beyond position: The emergence of wavelet-like properties in transformers. *arXiv preprint arXiv:2410.18067*, 2024. URL https://arxiv.org/abs/2410.18067.

Ning Shang, Li Lyna Zhang, Siyuan Wang, Gaokai Zhang, Gilsinia Lopez, Fan Yang, Weizhu Chen, and Mao Yang. LongRoPE2: Near-lossless LLM context window scaling. *arXiv preprint arXiv:2502.20082*, 2025. URL https://arxiv.org/abs/2502.20082.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2208.04933.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021. URL https://arxiv.org/abs/2104.09864.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL https://jmlr.org/papers/v9/vandermaaten08a.html.

Yu Wang, Sheng Shen, Rémi Munos, Hongyuan Zhan, and Yuandong Tian. Positional encoding via token-aware phase attention. *arXiv preprint arXiv:2509.12635*, 2025. URL https://arxiv.org/abs/2509.12635.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2309.17453.

Zhenhai Zhu and Radu Soricut. H-Transformer-1D: Fast one-dimensional hierarchical attention for sequences. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3801–3815. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.294. URL https://aclanthology.org/2021.acl-long.294/.

A NOTATION, TASKS, AND METRICS

Table 4 summarizes the notation used below.

Table 4: Notation used in the scan and language-model sections.

Symbol	Meaning
n, i, j	Sequence length, query token index, and key index; token positions start at 1 and BOS is 0.
$\Delta = i - j > 0$	Positive relative offset. Self and BOS are separate boundary cases.
$r = v_2(\Delta)$	Largest k for which 2^k divides Δ; trailing-zero count.
$s = \mathrm{msb}(\Delta)$	$\lfloor \log_2 \Delta \rfloor$; highest set-bit index or distance-scale bin.
t, T'	Zero-based scan iteration and chosen number of iterations.
$M, \circ, 1_M$	Finite state set (the carrier), associative binary operation, and identity element; together they form the monoid $(M, \circ, 1_M)$.
w_i^t, y_i	Intermediate dyadic-window product and final prefix product.
$e(m), P, f$	State codeword, nearest-codeword projection, and fixed sentinel.
c, b_c	Clipped dyadic cell and its learned per-head/per-layer LM bias.
$H, T, K_H(c)$	Test horizon, reference horizon, and count of positive offsets in a cell. T in the occupancy formulas is not a loop count.
λ, κ, s	Occupancy-correction strength, positional-interpolation factor, and YaRN scale, respectively. YaRN's s is distinct from the bit index in Eq. (1).

State and verification terms. A codebook is the finite set of state vectors $\{e(m) : m \in M\}$. Its elements are codewords or class centroids. A keep update retains the current window product when the left partner is absent. Raw inference passes continuous states between iterations without codebook projection. A sound enclosure contains every value allowed by the checker's stated arithmetic assumptions. A numerical audit tests computed values without this guarantee.

Language-model terms. Calibration here means compensation for changes in attention mass. It is distinct from probability calibration of model outputs. Addressing means selecting information at the relevant token position. The word deep refers to the table's scale axis, rather than network depth. NTK abbreviates neural tangent kernel, FT finetuning, NLL negative log-likelihood, and OOD out of distribution.

Other notation. ReLU is $\max(0, x)$ and GELU is $x\,[1 + \mathrm{erf}(x/\sqrt{2})]/2$. The t-SNE plots (t-distributed stochastic neighbor embedding) visualize high-dimensional states in two dimensions. EMA means exponential moving average, KL means Kullback–Leibler divergence, and CoT denotes chain-of-thought prompting. NC^1 is the class computed by polynomial-size, logarithmic-depth bounded-fan-in Boolean circuits. AC^0 uses constant depth and unbounded fan-in. An AC^0 reduction is computed by such circuits. These complexity classes concern representability, rather than measured training difficulty.

A.1 ALGORITHMIC TASKS

Parity reports the cumulative sum of bits modulo 2; for $(1, 0, 1, 1)$ the prefix labels are $(1, 1, 0, 1)$. $\mathbb{Z}_{60}$ reports cumulative sums of residue-class inputs modulo 60. S_5 uses the 120 permutations of five elements as states, with $(g \circ h)(u) = g(h(u))$ and the ordered prefix $x_1 \circ \cdots \circ x_i$. Permutation composition need not commute. **Carry** represents the effect of a binary digit pair on an incoming carry c: $f(c) = g \vee (p \wedge c)$, with $g = a \wedge b$ and $p = a \oplus b$. Its three functions are kill ($f = 0$), propagate ($f(c) = c$), and generate ($f = 1$). Composing them gives the carry transformation of a prefix. The final task label is the carry-out bit for incoming carry zero, $y_i = \mathbf{1}[\text{prefix transformation is generate}]$. Intermediate scan supervision instead uses all three state labels.

For Q evaluated sequences, let sequence q have n_q scored positions and define $I_{qi} = \mathbf{1}[\hat{y}_{qi} = y_{qi}]$. Then

$$\text{token accuracy} = \frac{\sum_{q,i} I_{qi}}{\sum_q n_q}, \qquad \text{exact-sequence accuracy} = \frac{1}{Q}\sum_q \prod_{i=1}^{n_q} I_{qi}.$$

Token accuracy can remain high when long-sequence exact accuracy is zero. The WAT and prompted-LLM probes that return a single final answer use final-answer accuracy, not all-prefix exact-sequence accuracy. Uniformly guessing one valid final answer succeeds with probability $1/2$ for parity and $1/120$ for S_5. The API probe also includes unanswered responses in its accuracy denominator, as specified in Appendix L. Three-seed summaries and seed-0-only cells are marked separately in Table 5. Zero observed errors is an empirical statement; an arbitrary-length guarantee additionally needs the certificate assumptions.

A.2 Language-model perplexity

For scored tokens $\mathcal{I}$, using natural logarithms,

$$\text{PPL} = \exp\left[-\frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \log p_\theta(x_i \mid x_{<i})\right].$$

Lower values indicate better average assigned probability on exactly those tokens. The main PG-19 grid scores the second half of each window, so the scored sets differ with window length. The same-target protocol instead fixes the last 2048-token block of each 65,536-token segment and varies available context. FineWeb-Edu validation uses a separate in-distribution dataset. Its local and training-run validation protocols score different token sets. For example, the 1.3B control has 9.34 under the local protocol and 9.94 under training-run validation.

A.3 Retrieval, variable tracking, and uncertainty

A **passkey** example places a five-digit key in filler and requests it later. Accuracy is the fraction of trials scored as recovering the key. The main comparison uses 5 insertion depths and 25 trials per depth (125 trials per length). The earlier protocol has 8 trials per depth (40 per length). **Variable tracking** follows chains of `VAR` assignments. A hop is one link needed to obtain the queried value. The reported 2/4/8/16-hop results probe a different capability from recovering a single verbatim key.

The **RULER subset** follows the four configurations described in Appendix U: single-needle-in-a-haystack S-NIAH-1 (a 7-digit value in an essay haystack), S-NIAH-2 (a repeated-noise haystack), S-NIAH-3 (a universally unique identifier, UUID, as the value), and multi-key MK-NIAH-1 (three distractor needles). Those tables use the scoring convention stated there, which differs from the custom passkey metric. There are 25 examples per task/length in that subset, rather than 125 in the headline passkey experiment.

For a binomial fraction $\hat{p} = k/N$, the Wilson 95% interval uses $z = 1.96$ and is

$$\frac{\hat{p} + z^2/(2N) \;\pm\; z\sqrt{\hat{p}(1-\hat{p})/N + z^2/(4N^2)}}{1 + z^2/N}.$$

It summarizes sampling uncertainty for a fixed model and protocol, excluding variation across training runs. Shared prompts make between-model comparisons paired.

B Proofs and scope of the theoretical results

B.1 Dilation shift (Lemma 1)

Multiplication by two shifts the binary expansion left by one bit. Both set-bit indices increase by one, which is canceled by replacing t with $t+1$. Coordinatewise clipping therefore preserves the equality. For the mask, a valid partner $i - 2^t$ becomes $2i - 2^{t+1}$; self and the sentinel map to $2i$ and 0. The existence condition $i > 2^t$ is unchanged under this dilation. □

B.2 UNIFORM SCAN (THEOREM 1)

The proof is a finite-state construction in exact real arithmetic, with sufficient width and a ReLU MLP. It is not a claim about the exact values learned by the GELU/bf16 experimental models.

State, read buffer, and normalization. Choose a canonical one-hot representation for each element of M and one distinct sentinel symbol. Reserve separate residual-stream coordinates for this state, for an initially zero attention-read buffer, and for two anchors that are always 0 and 1. All canonical codes have the same mean and variance. LayerNorm on canonical codes is therefore an affine map whose action can be absorbed into the value map. Attention writes a state-only value into the read buffer and leaves the state and anchors unchanged.

There are finitely many possible post-attention patterns: $|M|^2$ combine patterns, $|M|$ keep patterns, and the fixed sentinel pattern. Hard selection provides the partner code or sentinel code directly. For softmax, choose positive, fixed mixture coefficients so the sentinel coefficient differs between the combine and keep cases. For a fixed self code, the nonzero partner coefficient distinguishes the possible partner codes; the sentinel coordinate distinguishes a missing partner even when the valid partner would equal self. Thus these finite patterns are distinct. This is an existence choice of mixing coefficients, not a requirement that all learned logits satisfy it.

The pre-MLP LayerNorm does not collapse these patterns. Equality of two LayerNorm outputs implies that the unnormalized inputs differ by a positive rescaling and a constant shift (with fixed nonzero affine gains). The two unchanged anchor coordinates force that scale to one and the shift to zero. Consequently the normalized patterns remain distinct.

An explicit finite interpolation argument. Let the distinct normalized patterns be $z_1, \dots, z_N$, with $N = O(|M|^2)$. Choose a linear functional u such that the scalars $q_k = u^\top z_k$ are all distinct; the excluded choices form a finite union of hyperplanes. Order these scalars and assign at each q_k the desired residual increment: replace the state by the correct product/keep code, clear the read buffer, and leave the anchors unchanged. A continuous piecewise-linear vector function interpolates these prescribed increments. Its representation as an affine function plus a sum of hinges $(q - q_k)_+$ is a two-layer ReLU network of width $O(N)$. This supplies a lookup on the entire finite pattern set without assuming that each pattern is linearly separable from all the others. Residual addition yields canonical states again. Because those states have a common mean and variance, the final loop LayerNorm can be parameterized to act as the identity on them. The sentinel transition is assigned to keep its canonical value.

Window invariant and readout. Initialize token i with the code of x_i. Before loop t, assume it represents the ordered product from $\max(1, i - 2^t + 1)$ through i. When $i > 2^t$, the left partner represents the adjacent preceding window, so the MLP's ordered product is Eq. (2). When the partner is missing, the current window already reaches position 1 and is kept. This proves the invariant by induction. After $\lceil \log_2 n \rceil$ loops, every window is a full prefix, and the linear decoder maps its canonical state to the required answer. A transition monoid gives the regular-language consequence; the cited S_5 completeness result concerns an external input reduction rather than a learned reduction inside DST. □

B.3 FINITE TRANSFER AND PROJECTION (THEOREM 2, COROLLARY 1)

For codebook initialization, the preceding window induction applies verbatim when the two effective maps satisfy Eq. (4). The sentinel contribution and fixed attention mixtures are part of those maps, not omitted arguments of an otherwise content-dependent attention operator. Projection converts a correct nearest-centroid decision into exact equality to the required codeword. A base-case check must cover actual learned embeddings when they are not already codewords; thereafter the induction stays in the finite codebook. Length one with zero loops requires the corresponding input/readout check. A live sentinel instead creates a potentially loop-dependent family of maps and needs its own invariance or loop-index audit. Finally, finite-precision projection is covered only when the specified arithmetic error enclosure cannot cross any decision boundary. A decoded finite audit without those enclosures is not by itself a sound floating-point certificate. □

B.4 A PRECISE SELF-READ OBSTRUCTION

Proposition 2 (Self-read obstruction for a state-only scan interface)**.** *Consider a canonical per-loop parity scan: each state is only $e(m)$, $m \in \mathbb{Z}_2$, and the same update must implement both a keep step and a combine step. Suppose its only inputs are that state and normalized attention reads whose value vectors depend only on state, with no sentinel, boundary flag, role-tagged value, loop/index feature, or other side channel. If a keep case reads only self and a combine case reads self and an equal partner, those two inputs cannot be distinguished. This interface cannot implement both required transitions exactly.*

Proof. Take $m = 1$. Any normalized weighted average of one or several copies of the same value vector equals that vector, regardless of logits and number of heads. The keep and equal-partner cases therefore present the same self state and attention reads to the shared update. The required outputs are $e(1)$ and $e(1 \oplus 1) = e(0)$, respectively, a contradiction. □

The requirement is correctness of the canonical *per-loop* scan, not just final correctness at a selected depth. For example, correctness at $n = 1$ and zero loops does not force a recurrent update to be the identity. Boundary flags, additional recurrent state, role-specific values, or unnormalized aggregation can invalidate the indistinguishability premise. The sentinel resolves this ambiguity in the specified interface.

B.5 BOUNDED-LOGIT RESOLUTION

Proposition 3 (Bounded-range selection)**.** *Suppose one target is selected by softmax among $K \geq 2$ keys and the total logit range is at most D. Then non-target mass is at least*

$$\frac{(K-1)e^{-D}}{1+(K-1)e^{-D}}.$$

Consequently leakage at most $\varepsilon \in (0,1)$ requires $D \geq \log((K-1)(1-\varepsilon)/\varepsilon)$. If position logits are $g(\rho(\Delta))$ with feature diameter B and γ-Lipschitz g, their range is at most γB; an additional content-logit range at most C gives $D \leq \gamma B + C$. For strictly negative-slope ALiBi position logits alone, any non-nearest target with a nearer available key has probability less than one half.

Proof. Let $m_j = \ell_* - \ell_j$ be each competitor's margin. The odds of leakage are $S = \sum_{j \neq *} e^{-m_j}$. Each $m_j \leq D$, so $S \geq (K-1)e^{-D}$; leakage is $S/(1+S)$. The displayed condition follows by rearrangement. The Lipschitz and content bounds add. For the ALiBi-only case, a nearer key has a strictly larger logit than the target, so that competitor alone has odds greater than one. □

This is a bound on an individual selection operation with bounded scores, not a general impossibility theorem for RoPE, FIRE, ALiBi, or trained LMs. In particular, content can change the ranking. A uniform margin of order $\log(K/\varepsilon)$ is sufficient but need not be necessary for every competitor: the obstruction is the aggregate crowding within bounded range, not just the distance to an adjacent offset.

Fixed rotary readouts and recurrence. For fixed rotary frequencies, the position-only features form an orbit on a compact torus. Finite covers and the pigeonhole principle give arbitrarily late returns near the identity (exact returns for a finite orbit). For a *fixed continuous* scalar readout, sufficiently close returns yield competitors with arbitrarily small margin to a fixed target. Once such a competitor lies in context, leakage approaches at least one half. This recurrence argument assumes a fixed continuous readout. A family of increasingly sharp, length-dependent readouts falls outside that assumption. DST avoids growing competition through its mask.

B.6 EXTRA ITERATIONS

Proposition 4 (Idempotence)**.** *In the construction of Theorem 1, or under the premises of Theorem 2, every $T' \geq \lceil \log_2 n \rceil$ gives the same output.*

Once $2^t \geq n$, all token positions use the keep map, which fixes every codeword. The empirical loop-count check is separately reported for $T \in \{14, 16, 24, 40\}$ at $n = 16{,}384$; it does not test every integer in that range. □

B.7 Occupancy correction (Proposition 1)

For $c \in \mathcal{C}_+$, linearity of expectation gives

$$\mathbb{E}Z_H(c) = e^{b_c + \delta_H(c)} K_H(c)\mu_c, \qquad \mathbb{E}Z_T(c) = e^{b_c} K_T(c)\mu_c.$$

No independence among keys is needed for this equality. Matching the ratio for each pair c, d cancels b_c, b_d, μ_c, μ_d and requires

$$\delta_H(c) + \log \frac{K_H(c)}{K_T(c)} = \delta_H(d) + \log \frac{K_H(d)}{K_T(d)}.$$

Thus this quantity is the same a_H for all common populated cells; conversely that equality preserves all the required ratios. The uniqueness is only up to this common offset and only inside the specified class. □

Why normalization changes the claim. In general $\mathbb{E}[Z_c / \sum_d Z_d] \neq \mathbb{E}Z_c / \sum_d \mathbb{E}Z_d$. For example, take $Z_d = 1$ and let Z_c be $1/2$ or $3/2$ with equal probability. Then $\mathbb{E}Z_c = \mathbb{E}Z_d = 1$, but the expected normalized mass of c is $(1/3 + 3/5)/2 = 7/15$, not $1/2$. The proposition is therefore deliberately about ratios of first moments, not ratios after a random normalization.

Support and alternative geometries. If $K_T(c) = 0 < K_H(c)$, the logarithmic ratio is undefined: neither the formula nor its uniqueness fixes the treatment of that newly occupied cell. Masking, smoothing, or assigning a reference count would be additional method choices, not consequences of the proposition. The same caveat applies to the choice of reference horizon for an expanded table. For singleton offset cells, common-cell occupancies are one and this particular count correction is zero. Newly available offsets still change support, and other corrections remain possible. For v_2-only cells, counts grow with horizon while their proportions are approximately $2^{-(k+1)}$ (with a clipped tail). Hence the leading count change is common across cells and cancels under softmax; finite-horizon edge effects need not vanish exactly. With a single corrected cell, a common shift is ineffective only relative to other keys in that same corrected support; separately treated self/BOS keys require care. RoPE's content-dependent rotated inner products are outside the additive-cellwise class, not ruled out by it.

C Complete algorithmic results

Tables 5 and 6 give the complete baseline comparison for parity/carry and $\mathbb{Z}_{60}/S_5$, respectively. Seed counts and single-seed evaluations are identified in each table. Here 4k and 65k denote the reported power-of-two evaluation lengths 4096 and 65,536.

The parity curve is shown in Fig. 2.

D Experimental details

D.1 DST block and attention support

Figure 4 expands the masked, content-free DST of §2. The same learned block updates all token states in parallel at each iteration. The loop shares parameters and changes only the partner offset 2^t. Readout produces labels for the existing positions.

Masked attention. Let $h_i^t \in \mathbb{R}^d$ be the state before iteration t, including h_0^t for BOS. For head $a \in \{1, \ldots, H_{\text{attn}}\}$, set $v_{i,a}^t = W_{V,a} N_1(h_i^t)$, where N_1 is the attention pre-normalization. Each head has width $d_h = d/H_{\text{attn}}$. The head learns three scalar logits $\beta_a^{\text{partner}}, \beta_a^{\text{self}}, \beta_a^{\text{BOS}}$. For $i \geq 1$,

Table 5: Exact-sequence accuracy, parity and carry. Train $n \leq 64$, 32 sequences per length; averages over runs where specified.

	Parity ($\mathbb{Z}_2$)				Carry (addition)			
n	64	512	4k	65k	64	512	4k	65k
DST (ours)	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
DST + projection	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RoPE	0.72	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00
ALiBi	0.29	0.00	0.00	0.00	**1.00**	**1.00**	**1.00**	0.97
NoPE	0.00	0.00	0.00	0.00	0.88	0.00	0.00	0.00
FIRE	0.71	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00
Looped-NoPE ($T{=}n$)	0.72	0.00	0.00	—	**1.00**	0.00	0.00	—
LSTM (sequential)	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Measured 65k baseline cells use seed 0 (32 sequences). Dashes mark unmeasured Looped-NoPE cells with $T = n$; the DST+projection $n{=}64$ cells are exact (1.00) for each of the three seeds. FIRE and Looped-NoPE on carry/$\mathbb{Z}_{60}$/S_5: seed 0, matched budget (20k steps, width matched to that task's baselines), Looped-NoPE with the same auxiliary scan supervision as DST.

Table 6: Exact-sequence accuracy, $\mathbb{Z}_{60}$ and S_5. Train $n \leq 64$, 32 sequences per length; averages over runs where specified.

	$\mathbb{Z}_{60}$				S_5			
n	64	512	4k	65k	64	512	4k	65k
DST (ours)	**1.00**	**1.00**	0.02 $\pm$0.03	0.00	**1.00**	0.69 $\pm$0.40	0.21 $\pm$0.29	0.00
DST + projection	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RoPE	0.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ALiBi	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NoPE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FIRE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Looped-NoPE ($T{=}n$)	0.00	0.00	0.00	—	0.00	0.00	0.00	—
LSTM (sequential)	**1.00**	0.70	0.02	0.00	**1.00**	**1.00**	**1.00**	0.94

Measured 65k baseline cells use seed 0 (32 sequences). Dashes mark unmeasured Looped-NoPE cells with $T = n$; the DST+projection $n{=}64$ cells are exact (1.00) for each of the three seeds. FIRE and Looped-NoPE on carry/$\mathbb{Z}_{60}$/S_5: seed 0, matched budget (20k steps, width matched to that task's baselines), Looped-NoPE with the same auxiliary scan supervision as DST. The raw S_5 and $\mathbb{Z}_{60}$ DST summaries use three training seeds. Values after $\pm$ are population standard deviations.

define

$$\ell_{ij}^{t,a} = \begin{cases} \beta_a^{\text{partner}}, & j = i - 2^t \geq 1, \\ \beta_a^{\text{self}}, & j = i, \\ \beta_a^{\text{BOS}}, & j = 0, \\ -\infty, & \text{otherwise.} \end{cases} \qquad p_{ij}^{t,a} = \frac{\exp \ell_{ij}^{t,a}}{\sum_{k \in S_t(i)} \exp \ell_{ik}^{t,a}}.$$

The attention output is

$$m_{i,a}^t = \sum_{j \in S_t(i)} p_{ij}^{t,a} v_{j,a}^t, \qquad A_i^t = W_O[m_{i,1}^t; \ldots; m_{i,H_{\text{attn}}}^t].$$

When $i \leq 2^t$, the partner is absent and the softmax normalizes only self and BOS. For the BOS query itself, the support is $\{0\}$. Thus hard routing fixes the readable positions, while learned softmax weights can mix their values. The gather implementation evaluates these mixtures without constructing an $n \times n$ score matrix.

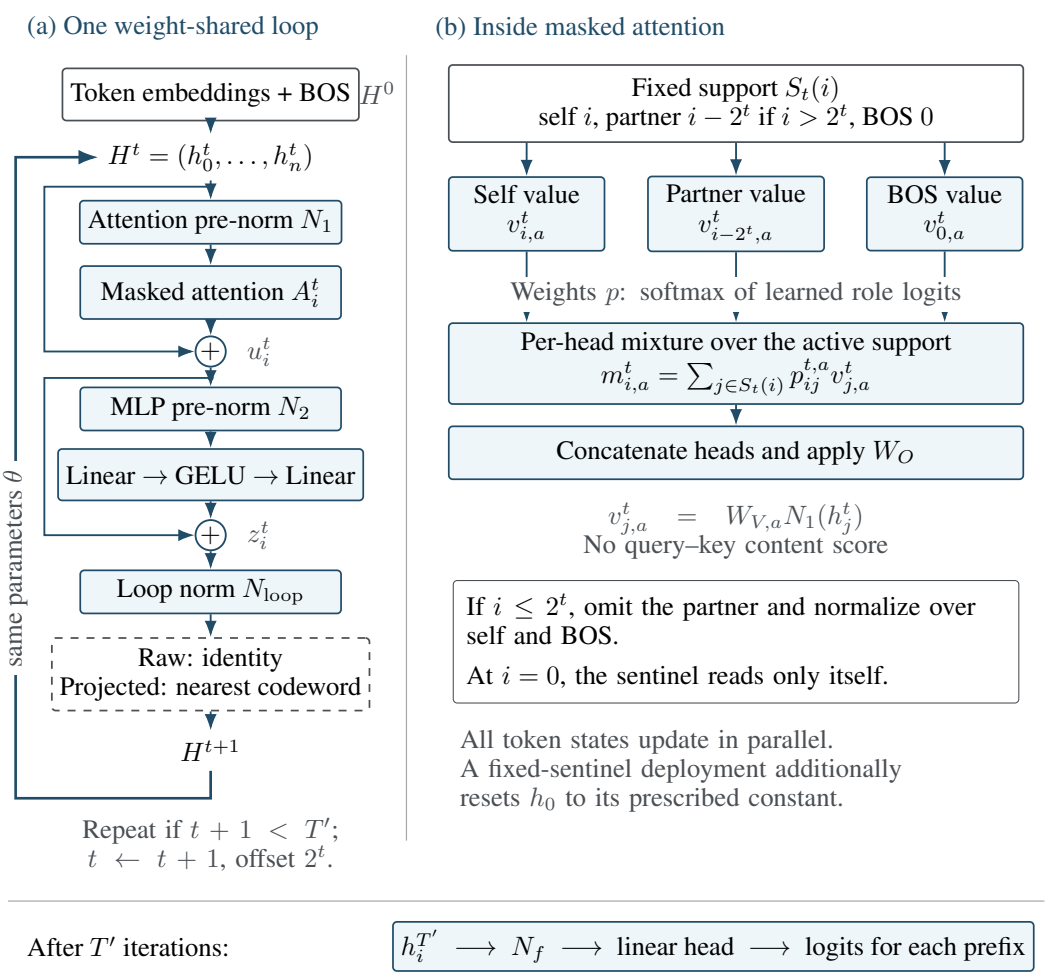

Figure 4: The looped DST and its attention sublayer. The dyadic support selects self, the left partner when present, and BOS. Learned per-head role logits directly determine their mixture. Both residual additions and the post-loop normalization are shown. Raw recurrence keeps the continuous output. Projected recurrence maps token outputs to the nearest codeword. Fixed-sentinel deployment additionally resets position 0 to f. All learned operators are shared across iterations. Readout uses a separate final normalization.

State update and readout. With the MLP $F(z) = W_2\,\mathrm{GELU}(W_1 z + b_1) + b_2$, one iteration is

$$u_i^t = h_i^t + A_i^t, \qquad z_i^t = u_i^t + F(N_2(u_i^t)), \qquad \bar{h}_i^{t+1} = N_{\text{loop}}(z_i^t).$$

Here N_2 is the MLP pre-normalization and N_{loop} is the post-loop normalization. The standard DST uses LayerNorm. The verification variants with constant-scale normalization are described in Appendix F. Raw inference sets $h_i^{t+1} = \bar{h}_i^{t+1}$. Projected inference instead sets $h_i^{t+1} = P(\bar{h}_i^{t+1})$ for $i \geq 1$, after both residual sublayers and loop normalization. A live sentinel follows its own update. The idealized fixed-sentinel construction sets $h_0^t = f$ at every loop. The certified learned model uses its input BOS in the base step and fixes the sentinel thereafter (Appendix E).

Learned token embeddings and BOS initialize the states. After T' iterations, the readout is $W_{\text{out}} N_f(h_i^{T'}) + b_{\text{out}}$, with final normalization N_f. It is also used for intermediate supervision, with a separate auxiliary head for carry. Certificate-specific initialization, decoding, and arithmetic conditions are in Appendix E.

D.2 Training and baseline configurations

Architecture. The scan model repeats one shared pre-LayerNorm block with 4 attention heads, MLP ratio 4, a BOS sentinel, and shared post-loop LayerNorm. Model widths are 128 for parity and carry, 256 for $\mathbb{Z}_{60}$, and 384 for S_5. Masked dyadic attention uses per-head logits $(\alpha, b_{\text{self}}, b_{\text{bos}})$.

Training. We use batches of 256, AdamW without weight decay, and bf16 autocast. The learning rate rises to 10^{-3} over 500 warmup steps, then follows a cosine schedule to 10% of that value. Runs use 15k–60k steps, sequence lengths $n \sim U[2, 64]$, loop count $T = \lceil \log_2 n \rceil + U[0, 10)$, and state noise $\sigma = 0.05$. Every loop receives supervision from its window-product label through the output head. Carry uses an auxiliary 3-way head for these labels.

Evaluation and codebooks. Each length uses 32 sequences and $T = \lceil \log_2 n \rceil$. The reported evaluations use dense attention through $n = 8192$ and the $O(n)$ gather path beyond, with a threshold of 4096 for projection runs. The supplied masked-model implementation selects the gather path when the sequence including BOS exceeds 4096. Empirical codebooks average states from 20 batches of 64 sequences at $n = 64$, grouped by window value. Projection maps each post-loop state to its nearest centroid. The certificate extractor uses a separate codebook from 40 batches of 32 sequences at length 64, with seed 999.

Baselines and hardware. Baselines are depth-6 transformers of matched width with RoPE, ALiBi, NoPE, or FIRE, a Fan et al.-style looped-NoPE model with input injection and $T = n$, and a 2-layer LSTM. All runs use a single RTX 4090.

E Certificate details and numerical transition audits

Certified transition system. Theorem 2 reduces correctness to initialization, combine/keep transitions, and decoding. In the specified float64 deployment, token states are projected onto a fixed codebook after each loop. Loop 0 reads the initial BOS embedding. Subsequent loops read the fixed sentinel b^*, obtained from its keep iteration with residual 3.6×10^{-15}. Stored effective maps absorb attention mixtures and value/output projections. Together with embeddings, normalization coefficients, MLP weights, codebook, and decoder, these constants define the complete update. All maps remain fixed during inference.

Arithmetic and decision bounds. A midpoint–radius enclosure records a center and a bound on its error. For unit roundoff u, the accumulation factor is $\gamma_k = ku/(1 - ku)$, with $ku < 1$. Matrix products, normalization, division, and square root use IEEE-754 forward-error bounds. For GELU, the error function $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-z^2}\, dz$ is enclosed by a rational Taylor grid with an alternating-series remainder, exact integer arithmetic, and outward conversion to float64. Monotonicity and an $\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$ tail bound beyond $|x| = 6$ complete the enclosure. The arithmetic contract requires runtime nonlinear evaluations to stay within these enclosures. The reference routine checks every call and replaces any out-of-enclosure library value with a value inside the certified interval.

Projection minimizes float64 squared Euclidean distances, summed over coordinates, with ties resolved to the lowest codeword index. A certified distance margin bounds the nearest incorrect-centroid distance minus the correct-centroid distance after enclosure error. If $m > 0$ is this margin and S_t bounds the squared distance to the correct centroid, the sufficient rounding-aware decision condition is $m^2(1 - \gamma_{d+3}) > 2\gamma_{d+3} S_t$. Final normalization and the output head are enclosed separately on all codewords and input embeddings.

Finite checks. The base enumeration covers $|\Sigma|^2 + |\Sigma|$ cases from input embeddings and the initial BOS, where Σ is the input alphabet. The transition enumeration covers $|M|^2$ combine and $|M|$ keep cases from codewords and b^*. For S_5, both have 14,520 cases, with minimum margins 9.830 and 10.540. The certificate-trained parity checkpoint has margins 4.582 and 4.416. Maximum componentwise output radii are 1.1×10^{-3} and 2×10^{-5}, respectively. These radii are ℓ_∞ quantities already accounted for in the Euclidean distance margins.

The enclosed readout has minimum logit margins 22.323 for parity and 25.829 for S_5. On unprojected input embeddings the margins are 23.440 and 2.091, covering the zero-loop case at $n = 1$. The reference implementation instead uses at least one loop at $n = 1$, whose keep transition is covered by the base check. The projection rounding condition has over 10^{12} multiplicative headroom on both checkpoints. With the specified constants and arithmetic, the finite checks give every prefix product at arbitrary length by induction.

Additional transition audits. A separate live-sentinel float64 audit covers 566,280 transitions through $t < 40$, corresponding to $n \leq 2^{40}$, without rounding-error enclosures. The earlier codebook audit checks 259,200 combine and 2,160 keep cases for $t < 18$, with minimum nearest-centroid margin 10.5. Perturbing the states within measured radii gives zero failures in 518,400 combine trials. Fixed-sentinel projected evaluation also has zero token errors on S_5 at 4096 and 65,536. These audits complement the specified float64 certificate. The empirical bf16 GPU evaluations and raw-inference analyses use distinct execution paths, described in Appendices D and F.

F Raw inference: stability and verification bounds

Section 4 tests the premises of Theorem 2 on trained models. Here we examine bounds and training interventions for raw inference using float64 computations without directed rounding. The results identify obstacles to a complete invariant-set certificate.

Local contraction. The initial model has maximum Jacobian spectral norm 1.43 across audited loop indices and combine/keep cases. Its keep update has spectral radius $\rho \approx 1.05$. Penalizing directional amplification $\|\Phi(h+\Delta) - \Phi(h)\|_\infty / \|\Delta\|_\infty$ produces models with zero observed errors at 65,536, maximum spectral norm 0.88, and $\rho(J_{\text{keep}}) = 0.84$ across cases and $t < 25$. Stronger penalties incur token error rates of 10^{-4}. These eight training variants measure local stability alongside accuracy.

Normalization and transition bounds. Interval and zonotope bounds for LayerNorm can inflate by 10^{6+} because the enclosing regions include states with very different variances. StaticLN replaces the variance-dependent scale with a trained constant after mean-centering. The resulting models retain the observed accuracy and admit the factorization $\Phi(x) = Ax + b + C\,\mathrm{gelu}(Dx + e)$, with a fixed sentinel and concatenated self/partner input x. The trained weights determine A, C, D, b, e. For input center c, target center a, and Jacobian blocks J_1, J_2, a joint Euclidean input ball gives

$$\|\Phi(c+\delta) - a\| \leq \|\Phi(c) - a\| + \rho\|[J_1, J_2]\|_2 + \mathcal{R}(\rho), \qquad \|\delta\|_2 \leq \rho.$$

The first term is drift. Independent radius-ρ input balls allow concatenated norm $\sqrt{2}\rho$, which must be reflected in the bound. One verifier-regularized checkpoint has drift 0.75, joint-ball linear ratio 0.78, and certified decode margin 10.3 at the measured radii. These component bounds still leave the nonlinear term to control.

The nonlinear remainder. For preactivation perturbation $v = D\delta$, the GELU remainder satisfies $\mathcal{R} = \|Cr\|$ and $|r_i| \leq \kappa_i |v_i|$. Here κ_i is the range of gelu' over the reachable interval of unit i. Three enclosures (per-coordinate grid, mean-value spectral, and rank-one sum) give $\mathcal{R} \approx 16$ where the remaining budget is approximately 0.4. Units crossing the strongly curved part of GELU dominate this bound. We therefore train against a differentiable surrogate and verify the full bound separately.

Curvature-regularized finetuning. We test whether reducing this surrogate can preserve long-sequence accuracy. All eleven runs start from the same contraction-trained parity model, labelled start in Fig. 5. It was trained from scratch for 25,000 steps with batch size 256 and learning rate 10^{-3}, and has zero token errors in the 65,536-token evaluation. All finetunes retain the dyadic routing mask during training and evaluation.

Let $\mathcal{S}$ contain the two keep cases and four ordered combine cases for parity states in $\{0, 1\}$. At radius $\rho = 1$, the training surrogate is

$$R = \frac{1}{6} \sum_{s \in \mathcal{S}} \sum_i \widehat{\kappa}_{s,i} \, \|C_{s,:,i}\|_2 \, \|D_{s,i,:}\|_2.$$

Here C_s, D_s are the matrices in the one-step factorization above for case s. The quantity $\widehat{\kappa}_{s,i}$ is the sampled range of gelu' on a grid centered at preactivation $u_{s,i}$ with radius $\rho\|D_{s,i,:}\|_2$. Preactivations are evaluated at class-centroid and sentinel anchors. The sound verifier also accounts for grid spacing and selects the tightest of three enclosures. Thus R is a training statistic rather than a certified bound.

At optimization step k, the loss is

$$\begin{aligned}\mathcal{L}_k &= \mathcal{L}_{\text{scan}} + 3\,\mathrm{KL}(p_0\|p_\theta) + w_R(k)R \\ &\quad + w_{\text{scale}}[\mathrm{rms}(u) - 1.05r_0]_+^2, \\ \mathcal{L}_{\text{scan}} &= \frac{1}{T}\sum_{t=1}^{T}\mathrm{CE}\big(\mathrm{head}(h_t), y_t\big).\end{aligned}$$

The scan term averages cross-entropy over T loop iterations against the corresponding window labels y_t. The distributions p_0 and p_θ are the final-loop predictions of the frozen starting model and the finetuned model. The scale penalty uses $[z]_+ = \max\{0, z\}$ and the mean root-mean-square preactivation, with initialization value r_0. The curvature weight is zero for the first 500 updates, then increases quadratically over 60% of the training budget towards an initial cap $w_R^{\max}$, with a 10^{-4} floor after the delay. Every 100 updates, the anchors are refreshed by an exponential moving average with weights 0.9 and 0.1 on the previous and newly estimated values.

Training and evaluation. Updates use fresh batches of 128 sequences with lengths uniform on 2–64. Loop jitter adds an integer uniform on 0–9 to $\lceil \log_2 n \rceil$. We use AdamW with zero weight decay and gradient clipping at 1.0. The accuracy guard, when enabled, evaluates eight fresh length-4096 sequences every 250 updates and halves the curvature cap below token accuracy 0.999.

Table 7 evaluates one final-step checkpoint per configuration, with one training seed and freshly estimated anchors. All rows use the same 32 length-$65{,}536$ sequences and an evaluation seed separate from training. Inference uses continuous states and a live sentinel. Integer error counts distinguish accuracies that nearly coincide in Fig. 5.

Results. Runs i and j reduce R from 44.09 to 0.41 and 0.47, respectively, with zero token errors in the shared evaluation. Thus a roughly hundredfold reduction of the surrogate is compatible with the observed long-sequence accuracy. Matched runs d and k differ only in whether the guard is enabled: d makes one token error, whereas k makes $1{,}008{,}159$ despite attaining a smaller surrogate (0.35 versus 0.55). This paired comparison supports the guard for the tested configuration. Run g nevertheless loses accuracy despite 23 reductions of the cap. These single-seed measurements establish accuracy on the tested inputs, while the all-input guarantee still requires the invariant-set checks.

State-radius constraints. A separate verifier-regularized model requires radius approximately 3 to cover its observed within-class state spread. At that radius the full bound exceeds the radius by about 12, and drift 0.75 exceeds the contraction budget $(1-\sigma)\rho \approx 0.6$. Compactness regularization reduces 65k accuracy to 0.99 at weight 0.3 and to 0.88 at weight 1.0 (1500 updates, learning rate 10^{-4}, task and KL losses retained). Short-sequence probes remain correct. These are separate compactness tests, illustrating a tradeoff between state coverage and long-sequence accuracy.

Box enclosures and reachable states. For a separate checkpoint with $R = 0.34$ and zero observed errors at $65{,}536$, the keep fixed-point residual is 7×10^{-7}. One-step amplification is 0.34 on sampled states, whereas a sign-aligned corner of the radius-0.61 axis-aligned box expands by 4.74 in the linear attention stage. This rules out that stagewise box enclosure. Later stages can still contract intermediate expansion, so full-loop and correlation-aware regions remain alternatives.

Scope of verification. Appendix E gives the projected fixed-sentinel certificate. For raw inference, the experiments establish local contraction and decode robustness at the audited states. A full invariant-set certificate remains open. It must control state spread and drift as well as the nonlinear remainder. The finetuning sweep shows that reducing the surrogate can preserve empirical long-sequence accuracy.

Table 7: Finetuning configurations and results for Fig. 5. All runs use the shared initialization and protocol described in the text. $w_R^{\max}$ is the initial curvature-penalty cap. Errors count incorrect tokens in the same $32 \times 65{,}536$ raw-inference evaluation. The starting model uses zero finetuning updates, and dashes mark inapplicable settings. R is rounded to two decimal places.

Run	Guard	Finetuning updates	LR	$w_R^{\max}$	w_{scale}	R	Token errors
start	—	0	—	—	—	44.09	0
a	off	4000	$3 \cdot 10^{-4}$	0.03	1	0.62	760,553
b	off	4000	$3 \cdot 10^{-4}$	0.01	1	0.42	55,068
c	off	4000	10^{-4}	0.03	3	0.33	1,008,161
d	on	6000	$5 \cdot 10^{-5}$	0.02	1	0.55	1
e	on	6000	$5 \cdot 10^{-5}$	0.005	1	0.62	831
f	on	6000	$2 \cdot 10^{-5}$	0.02	0.3	1.72	19
g	on	8000	$5 \cdot 10^{-5}$	0.05	1	0.88	1,008,395
h	on	10000	$5 \cdot 10^{-5}$	0.01	1	0.57	589,203
i	on	10000	$3 \cdot 10^{-5}$	0.014	1	0.41	0
j	on	12000	$5 \cdot 10^{-5}$	0.008	1	0.47	0
k	off	6000	$5 \cdot 10^{-5}$	0.02	1	0.35	1,008,159

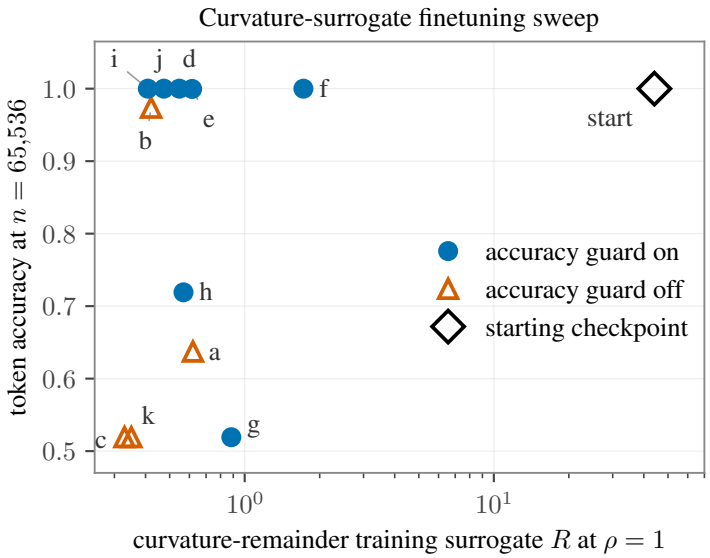

Figure 5: Curvature-regularized parity finetuning. The horizontal axis is the training surrogate R at $\rho = 1$. The vertical axis is raw token accuracy on the shared $32 \times 65{,}536$ evaluation. Blue circles use the accuracy guard, orange open triangles omit it, and the open diamond is the starting model. Letters identify the configurations in Table 7. Its integer error counts resolve accuracies close to one.

G Anatomy of failure in learned looped models

The following parity variants use matched data and training budgets.

No sentinel. Across four configurations, in-distribution accuracy plateaus at 82–98%, consistent with the self-read obstruction in Proposition 2.

Learned dyadic bias table. The free-table model fits the training data but fails at twice the training length. Its numerical feature indexing is described in Appendix J. Sharpness penalties disrupt opti-

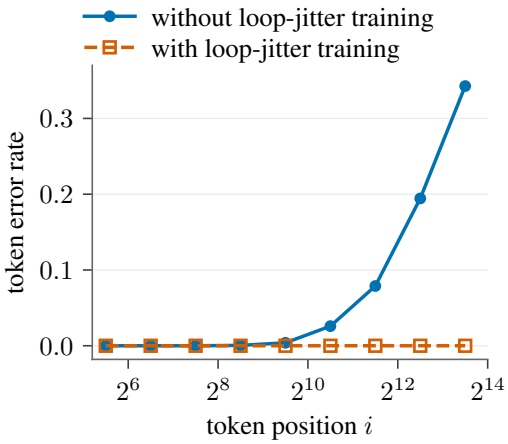

Figure 6: Token error rate versus position i in dyadic bins $[2^k, 2^{k+1})$, evaluated on eight parity sequences of length 16,384. The model trained without loop jitter accumulates errors at later positions. The model trained with loop jitter has zero errors in every plotted bin.

mization, and sparsemax retains a distributed read. Replacing the mixture by hardmax at evaluation reduces accuracy to chance. These outcomes characterize the tested implementation and training recipe.

Structural mask. The model is exact at 128, but token accuracy falls to 0.9978 at 512 and deteriorates at longer lengths.

Loop LayerNorm without content attention. Errors persist and increase with $\lfloor \log_2 i \rfloor$. The same position-dependent pattern appears in fp32 and bf16.

Loop jitter and state noise. These additions eliminate the observed errors through 2^{20} in this run. The raw recurrence has no formal fixed-point certificate.

Matched ablations. The following parity sequence accuracies use identical budgets and lengths $64/512/16{,}384$. Full recipe: $1.0/1.0/1.0$. Remove scan supervision: $0.00/0.00/0.00$ (in-distribution training also fails). Remove sentinel: $0.25/0.00/0.00$ (Prop. 2). Replace mask by the learned feature table: $1.0/0.00/0.00$ (Appendix J). Remove loop LayerNorm: $1.0/0.00/0.00$ (drift). Loop jitter and state noise stabilize recurrence: either alone gives $1.0/1.0/1.0$ on parity. Without both, accuracy decays with length (Fig. 6). Adding content attention to the full recipe also gives $1.0/1.0/1.0$. The content-free form supplies the fixed mixtures used in the theory.

H Supplied structure and learned transitions

Matched intermediate supervision. We give the fixed-depth RoPE baseline the same dyadic window targets per layer, with depth six covering training lengths. Looped-NoPE receives linear-window targets per iteration. RoPE reaches zero sequence accuracy at twice the training length with or without this supervision. Looped-NoPE reaches 0.05 at 128 and 0.00 at 256, compared with 0.68 and 0.00 without the intermediate labels.

Routing and representation. DST supplies the per-iteration support and learns the state-composition update and representations. The free-table and family-mask runs lose accuracy under length extrapolation. Appendix J describes their numerical feature indexing and the routing probes, including sparsification and sparsemax. The family contains at most 15 keys for $n \leq 8192$. Selecting the required offset 2^t at each iteration gives the successful scan. This is an explicit communication prior, analogous to a fixed convolutional access pattern.

Relation to language-model attention. Both settings expose changes in the number of keys assigned to a positional bias. Their successful interventions differ. The algorithmic models use precise per-scale routing and stable states, while occupancy correction improves LM perplexity under the tested conditions. Applying that correction to the small soft-table model leaves its extrapolation failure unresolved.

I ADDITIONAL ALGORITHMIC-REGIME AUDITS

Tree-structured baselines. WAT (Berezkin, 2026) uses a shared binary-tree GLU merge without attention. Run from its public code on the same final-answer protocol (3 seeds each), it fits parity perfectly in distribution (1.00 at $n \leq 64$) and is at chance (0.50) from $n=128$ on with its published learned positional embedding. Without that embedding, training accuracy is 0.87 and falls to chance by 16k. On S_5, accuracy stays at chance ($1/120$) even in distribution. We additionally run an LDRU-style baseline (Pert et al., 2026) head-to-head under the same budget and supervision: the Kogge–Stone gather hard-wired as a tensor shift (no attention), a learned combine MLP shared across levels, a learned identity vector at the left edge, and the same per-level scan supervision and loop jitter as DST. On parity it matches DST exactly (token accuracy 1.0000 at 65,536, consistent with Pert et al., 2026). On $\mathbb{Z}_{60}$ it is perfect in distribution but drifts at length ($0.99/0.84/0.54/0.24$ token accuracy at $1\text{k}/4\text{k}/16\text{k}/65\text{k}$, sequence accuracy 0 from 4k). On S_5 it fails much earlier (0.48 token accuracy at 256, near chance beyond 1k) and is imperfect even in distribution (0.20 sequence accuracy at 64). This shared-MLP scan adapts LDRU's topology rather than its published gated operator. The $\mathbb{Z}_{60}$ results show drift after accurate short-input computation. The S_5 model also has incomplete in-distribution learning. Comparison with projected DST evaluates the full cell and stabilization recipe.

Projection in the non-attention scan. We separately evaluate both models on the same 32 sequences at each length in $\{64, 1024, 4096, 16384, 65536\}$. Each codebook is constructed from 20 batches of 64 training-distribution sequences of length 64, grouped by per-loop window labels. For the shared-MLP scan, projection acts on the post-normalization recurrent state. The learned boundary identity is unchanged. At 65,536, token accuracy rises from 0.234 to 1.000 for this control and from 0.639 to 1.000 for DST. Both projected models have token and exact-sequence accuracy 1.000 at every evaluated length. Projection therefore benefits both tested cells. The comparison retains their respective architectures and training recipes.

Sequential and carry controls. The sequential LSTM is exact on parity and achieves 0.94 sequence accuracy on S_5 at 65,536. Its $\mathbb{Z}_{60}$ sequence accuracy falls from 0.70 at 512 to 0.00 by 8192, below DST with projection in this comparison. ALiBi reaches 8192 on random carry inputs. The retained-carry test at 4096 places one generate among the first four positions and uses propagate maps elsewhere, with zero incoming carry. ALiBi's reported token accuracy is 0.005, versus 1.0000 for DST.

A separate paired test compares GP^{n-1} and KP^{n-1} at $n \in \{64, 512, 4096\}$. Here G, K, and P denote generate, kill, and propagate. Each pair shares its propagate suffix. The two token encodings of P are sampled uniformly. On 32 pairs per length, DST has token and final-answer accuracy 1.000 for both families. ALiBi's final-answer accuracy is 0.000 for the generate family and 1.000 for the kill family at all three lengths. Its token accuracy on the generate family is 0.3149, 0.0396, and 0.0050, respectively. Both models are exact on the all-propagate control P^n, whose target is zero throughout.

Representation geometry. Appendix K describes the S_5 representation audit (Fig. 7). For parity, inter-class centroid distance is 23.5 and within-class spread is at most 5.9. Centroids for the same class across loop depths and boundary conditions are within 1.2. These measurements motivate codebook projection.

Loop-count robustness. At $n=16{,}384$ ($T^*=14$), $T = 13$ gives 0.75 token accuracy. The tested counts $T \in \{14, 16, 24, 40\}$ give 1.0000. The model performs the scan in exactly $\log_2 n$ levels and preserves accuracy at the additional tested loop counts, consistent with Proposition 4 under its stated hypotheses.

Training-length ablation. Training on $n \leq 32$ (instead of 64) still yields exact parity at 65,536 ($2048\times$) with raw inference. Training on $n \leq 16$ gives 99.1% token accuracy raw and exact accuracy with codebook projection, a $4096\times$ extrapolation from sixteen-token training sequences.

J Extracting structure from learned models

The routing probes use parity models trained on $n \leq 64$, either from scratch or from a free-table checkpoint, and evaluate through $n = 8192$. Selected probes also include S_5 and $\mathbb{Z}_{60}$.

Numerical feature indexing. These runs compute the scale index using floating-point logarithms on CUDA. For some power-of-two offsets, rounding below an integer places the offset in the preceding scale bin. At $\Delta = 8$ and $t = 3$, for example, the implementation uses $(0, -1)$ instead of the exact pair $(0, 0)$ in Eq. (1). The results below use this numerical map. Attention mass depends on key counts, content scores, and self/BOS terms. A softmax over table parameters alone measures a different quantity. Clipped diagonal corner cells also contain non-power-of-two offsets. We therefore report task accuracy for the routing probes.

Free-table extrapolation. The three tested soft-table models lose accuracy when the input length increases from 64 to 128. For S_5, token accuracy falls from 0.995 to 0.582, and exact-sequence accuracy from 0.81 to 0.00.

Top-cell sparsification. Restricting attention to each head's highest-valued table cell and finetuning for 2000 steps gives chance accuracy in the parity run. The selected cells use the numerical indexing described above.

Small-scale occupancy correction. The soft tabular model retains its length-generalization failure after the correction of §5 at $\lambda \in \{0.5, 1\}$.

Learning within a dyadic family. The support contains offsets $\Delta = 2^k$ together with self and BOS, while the mixture over these positions is learned. Parity reaches 100% training-distribution accuracy after 15k steps. At $n = 128$, token accuracy is 0.84 and exact-sequence accuracy is 0.00. Token accuracy is at chance beyond 512. The support contains at most 15 keys for $n \leq 8192$. On $\mathbb{Z}_{60}$ and S_5, the family-masked models also have low in-distribution accuracy: 0.32 and 0.20 token accuracy at $n = 64$ after 25–40k steps at two learning rates.

Sparsemax within the family. Sparsemax (Martins & Astudillo, 2016) permits one-hot selection. The parity run still fails to extrapolate to twice the training length. Under the family mask, S_5 training exhibits loss spikes after reaching 100% accuracy with both softmax and sparsemax.

These results characterize the routing recipes with the numerical map above. The effect of exact indexing on their extrapolation remains an open experimental question. Hard-routing DST selects its partner directly at integer offset 2^t.

K Codebook geometry

Construction and use. A codebook assigns one representative vector $e(m)$ to each symbolic state $m \in M$. After training, we collect non-sentinel post-loop states from short sequences and group them by their task-computable window product (Eq. (2)). For each state m, let $\mathcal{H}_m$ be the multiset of vectors with that label. We use their centroid and Euclidean projection:

$$e(m) = \frac{1}{|\mathcal{H}_m|} \sum_{h \in \mathcal{H}_m} h, \qquad P(h) = e\left(\arg\min_{m \in M} \|h - e(m)\|_2 \right).$$

The empirical extraction uses 20 batches of 64 sequences of length 64 and pools states across loop depths (Appendix D). The resulting vectors are fixed during evaluation. After each loop, projection selects the nearest codeword using only the current hidden vector. This prevents continuous state deviations from carrying over between iterations. The fixed-sentinel certificate separately uses

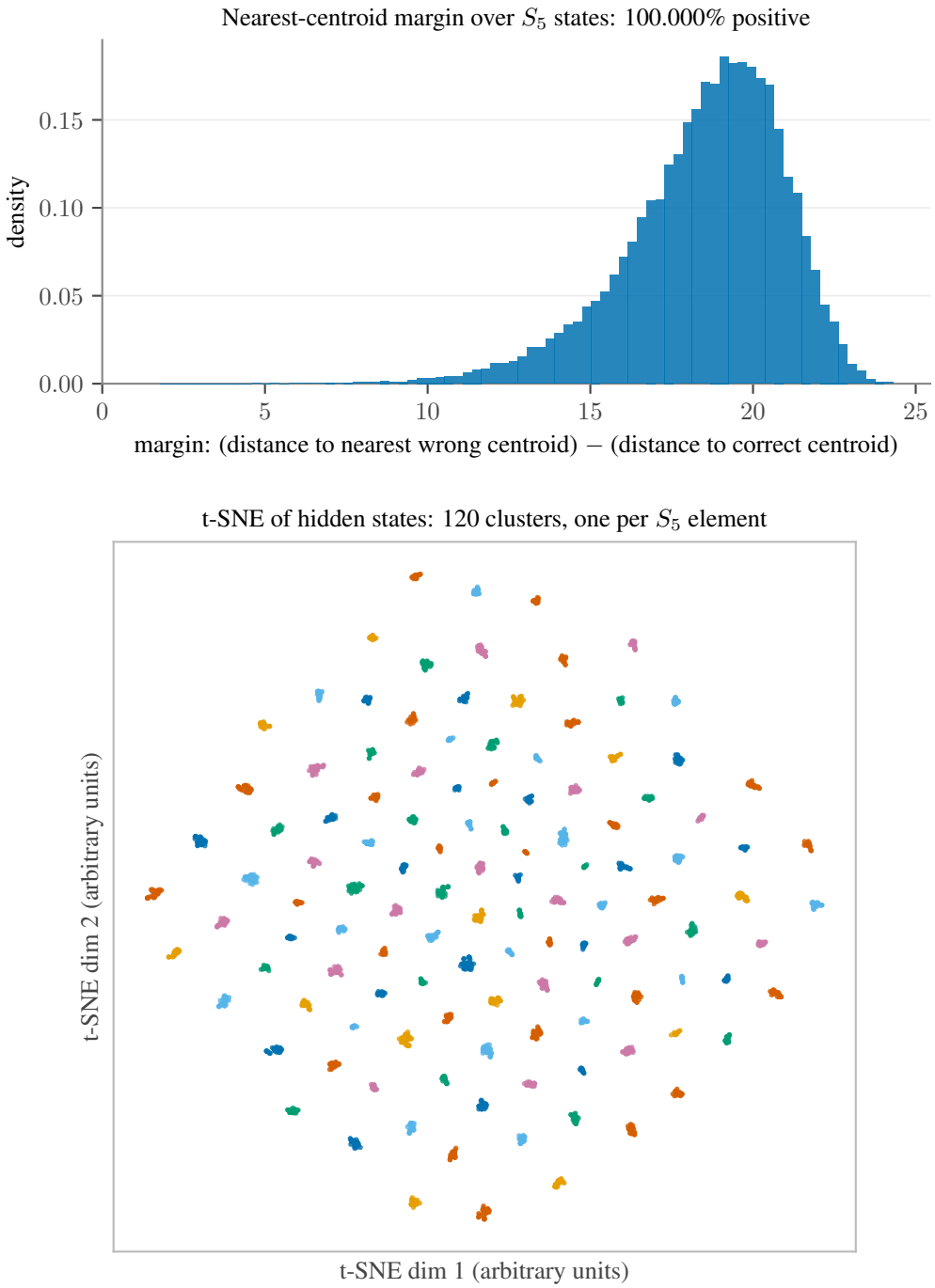

Figure 7: Learned state geometry on S_5 (28,672 post-loop states, all loop depths pooled, n=128). Top: the nearest-centroid margin (distance to the nearest wrong centroid minus distance to the correct centroid) is positive for 100.000% of states (min 1.78, median 18.9). Bottom: t-SNE shows 120 tight, separated clusters, one per group element; the six colors distinguish neighboring clusters and repeat across distant ones.

its pinned codebook and sentinel constant, with the extraction and verification conditions in Appendix E.

Representation audit. Figure 7 examines the learned S_5 geometry using a separate sample at length 128. For this visualization, states are passed through the final normalization, grouped by their window products, and compared with centroids computed from that same sample. The positive nearest-centroid margins describe separation in the hidden space. The t-SNE plot (van der Maaten & Hinton, 2008) shows a two-dimensional view of these states. The visualization is a representation diagnostic, whereas the projected-inference guarantee uses the finite checks in Corollary 1.

L PROMPTED-LLM STATE-TRACKING PROBE

We assess prompted state tracking on the algebraic tasks of §4. DST uses symbolic inputs and scan supervision. The API models receive textual task descriptions.

Protocol. We query six language-model families through OpenRouter. A parity prompt asks for the XOR of n bits. An S_5 prompt gives n permutations of $\{0, 1, 2, 3, 4\}$ in one-line notation (the digit at position x is $f(x)$) and requests $g = f_n \circ \cdots \circ f_1$, with f_1 applied first. Each task and length uses 16 fixed-seed instances shared across models. Requests contain one user message, use temperature 0, and ask for a labelled final answer within a requested generation limit of 32,000 tokens. Parity prompts also invite intermediate reasoning.

Scoring and response handling. For each model and length, accuracy is correct answers divided by scored responses. Three provider failures are excluded from 1,248 requests. Malformed answers (invalid permutations) and unanswered outputs count as unsuccessful. Tables 8–9 and Fig. 8 summarize the results. Of 125 responses without a final answer, 124 end with the API's length-limit status.

Table 8: Final-answer accuracy of prompted language models on 16 shared instances per task and length. † marks 15 scored responses after one provider failure. The outcome categories are defined in the text and counted in Table 9.

(a) Parity, n input bits

Model	16	32	64	128	256	512
DeepSeek v4-flash	1.00	0.94	1.00	0.75	0.62	0.50
Qwen 3.7-max	1.00	1.00	0.94	0.81	0.25	0.19
Kimi k2.5	1.00	1.00	0.94	0.94	0.50	0.38
GLM-5	1.00	1.00	0.69	0.62	0.38	0.31
Llama-4-Maverick	0.94	0.69	0.69	0.62	0.56	0.50
Grok 4.3	1.00	1.00	0.75	0.47†	0.56	0.47†

(b) S_5, n input permutations

Model	4	8	16	32	64	128	256
DeepSeek v4-flash	1.00	1.00	1.00	1.00	0.88	0.67†	0.00
Qwen 3.7-max	1.00	1.00	1.00	1.00	0.88	0.38	0.19
Kimi k2.5	1.00	1.00	1.00	0.94	0.75	0.19	0.06
GLM-5	0.94	1.00	1.00	0.69	0.81	0.12	0.00
Llama-4-Maverick	0.25	0.06	0.00	0.00	0.06	0.00	0.00
Grok 4.3	1.00	1.00	0.75	0.38	0.38	0.00	0.00

Table 9: Response outcomes across six services (96 requests per task and length). Parentheses count length-limit terminations. Provider failures are excluded from accuracy.

Task	n	Correct	Incorrect	Malformed	No final answer (length limit)	Provider failures
Parity	16	95	1	0	0	0
	32	90	6	0	0	0
	64	80	16	0	0	0
	128	67	22	0	6 (6)	1
	256	46	34	0	16 (16)	0
	512	37	30	0	28 (28)	1
S_5	4	83	12	1	0	0
	8	81	10	5	0	0
	16	76	14	6	0	0
	32	64	19	8	5 (5)	0
	64	60	30	1	5 (5)	0
	128	21	54	0	20 (20)	1
	256	4	47	0	45 (44)	0

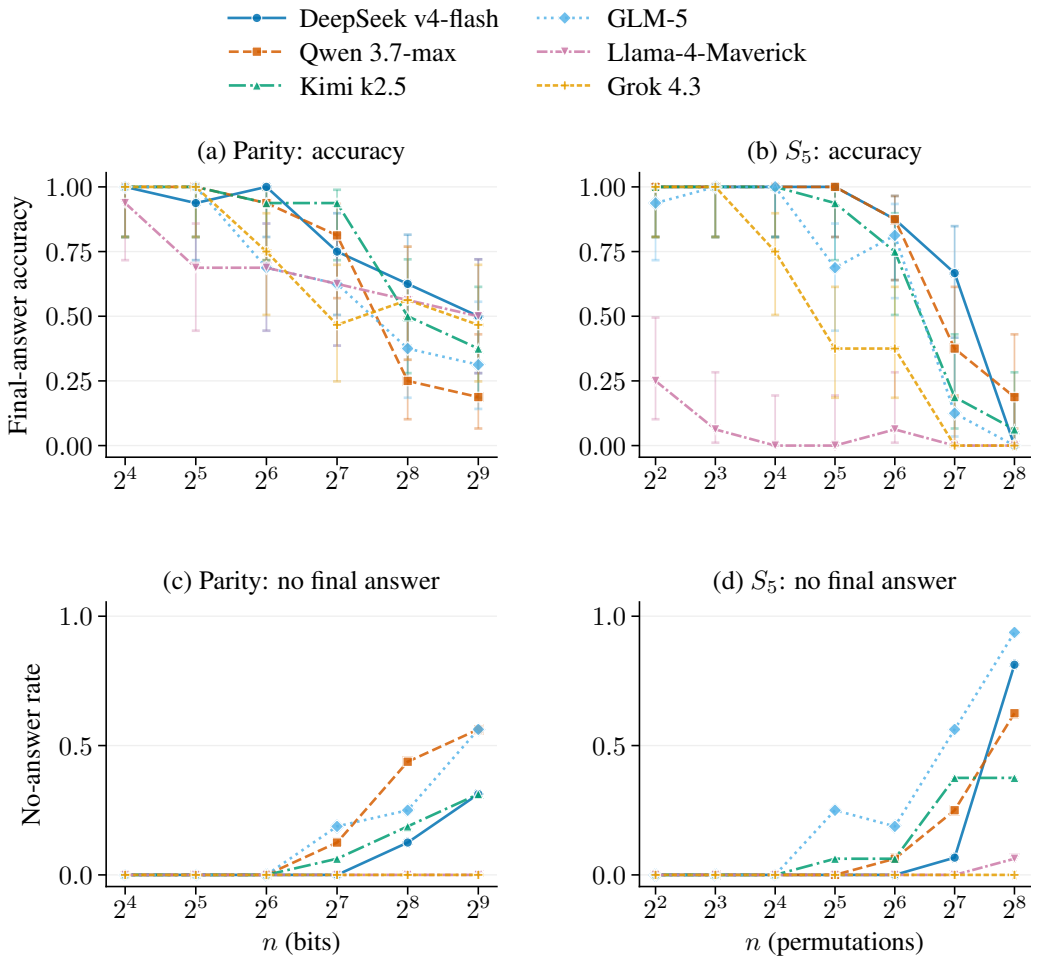

Figure 8: Prompted state tracking: final-answer accuracy with per-cell 95% Wilson intervals (top) and no-answer rate (bottom). Each cell uses 15 or 16 instances. Lines connect tested lengths.

M COMPLETE LM RESULTS AND EVALUATION PROTOCOLS

Table 10 gives the 1.3B perplexity grid. Table 11 holds target tokens fixed, and Table 12 reports the 125-trial passkey comparison. The numeric columns give evaluation context lengths. Every model starts from 2048-token pretraining. Table 12 contains only 16,384-token finetunes and the unadapted RoPE control. Full-budget deep-16k, joint dyadic+PI, and YaRN use 1.573B finetuning tokens. Their optimizer, learning rate, and data shard are specified in §6. The two † rows share the RoPE initialization, data order, optimizer, learning-rate schedule, and 196.608M-token budget. They differ in the positional-rescaling method. The reduced PI run is separate from the earlier PI finetune. Both use $\kappa = 16$ and a 16k maximum training context. These paired controls compare YaRN with PI. The same-target protocol scores the final 2048-token block of each 65,536-token segment, using the 2k result as each model's local reference.

Table 10: PG-19 perplexity after 1.3B / 21B-token pretraining at context 2048. The second half of each window is scored. The validation column uses one held-out FineWeb-Edu protocol. Occupancy correction uses $\lambda = 1$ unless shown otherwise. Evaluation-only transformations preserve the base model at context 2048. Finetuning contexts are shown in the row labels. Full-budget finetunes use 1.573B tokens. The † PI row uses 196.608M tokens and shares its training settings with the reduced YaRN control in Table 12.

Configuration	2048	4096	8192	16384	32768	val
RoPE (control)	17.8	68.3	290.2	401.4	480.4	9.34
+ YaRN (eval-time)	17.8	18.2	19.1	21.3	25.9	9.34
+ dynamic-NTK (eval-time)	17.8	18.7	28.0	60.6	131.1	9.34
RoPE + dyadic	17.5	62.0	325.5	411.3	478.1	9.33
+ occ. renorm.	17.5	51.8	250.4	317.8	350.2	9.33
($\Delta\%$ vs control)	-2	-24	-14	-21	-27	
NoPE + dyadic	18.4	60.5	269.9	759.1	1388.4	9.80
+ occ. renorm. (λ=0.5)	18.4	96.3	302.4	645.7	1064.1	9.80
Tail (2k FT, 1.5k updates)	17.4	62.9	330.3	417.4	480.2	9.25
Deep-16k (1.5k updates)	17.4	17.6	17.9	17.9	19.5	9.26
+ occ. renorm.	17.4	17.8	17.8	17.7	18.1	9.26
(replicate, seed 2027, +occ)	17.7	18.2	18.2	18.1	18.5	9.26
Deep-64k (1.5k updates)	17.8	18.5	19.0	18.8	19.0	9.27
+ occ. renorm.	17.8	18.3	18.6	18.5	18.6	9.27
Joint dyadic+PI (16k FT)	18.2	17.9	17.9	17.8	19.4	9.48
YaRN-16k (full budget)	17.9	17.6	17.5	17.5	17.6	9.31
PI-16k (reduced)†	19.4	19.1	19.0	18.7	19.8	9.85

Table 11: PG-19 perplexity on identical target tokens, varying the available context. All models were pretrained at 2048. YaRN-FT and deep-16k were finetuned through 16,384 tokens with the same full token budget. The evaluation-only YaRN row uses the pretrained RoPE checkpoint.

same-target ppl	L=2k (local)	4k	8k	16k	32k	65k
RoPE control	18.8	66.6	319	439	503	576
+ YaRN	18.8	17.9	18.8	21.4	26.5	36.0
YaRN-FT (full; dynamic scale)	18.6	17.4	17.2	17.4	17.5	19.3
Deep-16k FT	18.5	17.8	17.9	18.2	21.0	22.5
+ occ	18.5	18.0	17.9	17.9	18.3	18.3

Training trajectories. Across twelve matched validation checkpoints, dyadic validation loss is equal or lower. At 4096, its PG-19 perplexity is 9–14% lower across the reported stages. The RoPE control's 4096-token perplexity increases from 43.8 to 68.3 between 5k and 20k updates (Fig. 3).

Table 12: Passkey retrieval accuracy with 95% Wilson intervals from 125 trials per configuration and evaluation length. All models were pretrained at 2048 tokens. Every finetuned row uses a maximum training context of 16,384 tokens. Rows marked † are a matched YaRN/PI pair with 196.608M finetuning tokens each, one eighth of the main 1.573B-token budget. The separate joint dyadic+PI checkpoint supplies the headline retrieval result.

Configuration	2k	4k	8k	16k	32k
RoPE control	.62 [.53, .70]	.00 [0, .03]	.00 [0, .03]	.00 [0, .03]	.00 [0, .03]
PI-16k (κ=16)	.82 [.74, .87]	.78 [.70, .84]	.77 [.69, .83]	.55 [.47, .64]	.00 [0, .03]
Deep-16k + occ.	.70 [.61, .77]	.36 [.28, .45]	.18 [.13, .26]	.03 [.01, .08]	.00 [0, .03]
Joint dyadic+PI	**.92** [.86, .96]	**.81** [.73, .87]	**.94** [.88, .97]	**.75** [.67, .82]	.00 [0, .03]
YaRN-16k (full budget)	.84 [.77, .89]	.80 [.72, .86]	.60 [.51, .68]	.22 [.16, .30]	.00 [0, .03]
YaRN-16k (reduced)†	.81 [.73, .87]	.82 [.74, .87]	.78 [.70, .84]	.29 [.22, .37]	.00 [0, .03]
PI-16k (reduced)†	.84 [.77, .89]	.84 [.77, .89]	.81 [.73, .87]	.66 [.58, .74]	.00 [0, .03]

Extending the retrieval finetune. The $\kappa = 32$ joint variant uses the same 1500-update schedule with 32k long-context updates and reduced batch 64×2048. Its token budget is approximately one eighth of the full-budget runs, and embeddings and the head are frozen. It reaches passkey accuracy $0.27\,[.20, .36]$ at 32k and RULER S-NIAH-2 recall 0.72, with perplexity 18.8–18.9 through 32k. Its short-context passkey accuracy is 0.73, and retrieval fails at 65k. This comparison changes the finetuning range, interpolation factor, budget, and trainable parameters.

N RoPE-RESCALING CONFIGURATIONS

Evaluation-only rescaling. YaRN and dynamic-NTK are applied to the pretrained RoPE control with scale $s = n/2048$. Dynamic-NTK rescales the rotary base as $\theta \mapsto \theta s^{d_h/(d_h-2)}$ for head width d_h. Table 10 gives the common 2k–32k grid. At 65k, perplexity is 34.6 for YaRN and 220.9 for dynamic-NTK.

YaRN temperature. YaRN combines NTK-by-parts interpolation with attention-temperature scaling (Peng et al., 2023). We use ramp parameters $\alpha = 1$ and $\beta = 32$. We parameterize temperature directly at the attention-logit level with multiplier $1 + 0.1 \ln s$. This convention is held fixed during finetuning and evaluation, implemented by scaling the rotated query. The scale is $s = n/2048$ for evaluation-only rescaling and $s = \max(8, n/2048)$ for the finetuned baseline.

Finetuning settings. The full-budget YaRN run uses 1500 updates, with every twelfth update at 16,384 tokens and the rest at 2048. It matches the deep-16k token budget and context schedule. Optimizer, learning rate, and data shard differ. The reduced-budget YaRN/PI pair uses one eighth of the batch size, 196.608M tokens per run, and the same 2k/16k context schedule. Both reduced runs use the same RoPE initialization, data order, optimizer, learning rate, and trainable parameters. PI uses $\kappa = 16$, while YaRN uses $s = 8$ during finetuning.

Dynamic scaling beyond the 16k finetuning range is important: holding $s = 8$ fixed gives perplexity 97.9 at 32k for the reduced YaRN run. The dynamic rule gives 17.6 at 32k and 18.5 at 65k. The full- and reduced-budget YaRN runs differ by at most 0.1 perplexity on the measured length grid, although their retrieval differs (Table 12). At 131k, full-budget YaRN reaches 21.7, compared with 20.9 raw and 19.9 corrected for deep-64k. The latter reaches 20.9 raw at 262k, where the YaRN evaluation exceeds the available GPU memory. Same-target comparisons are in Table 11.

O LANGUAGE-MODEL TRAINING AND INFERENCE

Pretraining configuration (all 1.3B pretrains). The dense Llama-style decoder uses 24 layers, $d_{\text{model}} = 2048$, 32 standard multi-head attention heads, SwiGLU feedforward width 5504, RMSNorm, a 32k-token TinyLlama tokenizer, and tied input/output embeddings (1.28B parameters).

All runs use the same 21B FineWeb-Edu tokens (sample-100BT), stored as uint16 shards. Optimization uses AdamW ($\beta = (0.9, 0.95)$, weight decay 0.1), peak learning rate 3×10^{-4}, 2k warmup steps followed by cosine decay to 10% of the peak, and 20,000 updates at global batch 512 and context 2048 (1.05M tokens per update). We use bf16 autocast, activation checkpointing, and gradient clipping at 1.0. Control and RoPE+dyadic use 4×H100 each with distributed data parallelism and identical data order and seeds. NoPE+dyadic uses 8×H100 with different sharding of the same data. The dyadic bias adds one zero-initialized 9×9 table per head per layer (+0.005% parameters). FlexAttention computes attention, with the table replicated across batch and row-group tiles to distribute gradient atomic updates. The 150M, 1.3B, and 3B implementations construct feature indices on the CPU and transfer the integer buffers to the GPU. These buffers agree with the integer definitions through offset 262,144.

Inference implementation and efficiency. For evaluation horizon n, each layer combines its dyadic table, optional learned tail, and global occupancy correction into a float32 offset vector:

$$b_h(\Delta) = B_{h,c(\Delta)} + \tau_h(\Delta) - \lambda \ln \frac{\widetilde{K}_n(c(\Delta))}{\widetilde{K}_T(c(\Delta))}.$$

Here $c(\Delta)$ is the clipped dyadic cell, and τ_h is zero except in the tail variant. The self offset uses the cell for $\Delta = 1$. This offset-only implementation uses global occupancy correction. Per-query correction requires a query-dependent bias. The implementation rebuilds the vector at each forward call over its configured offset range $0, \ldots, n_{\max}$, costing $O(Hn_{\max})$ per layer for H heads. The benchmark uses $n_{\max} = 131{,}072$. The occupancy terms are cached by horizon and correction strength.

The forward-only Triton kernel adds $b_h(i-j)$ to $q_i k_j^\top / \sqrt{d_h}$ in a FlashAttention-2-style tiled forward (Dao, 2024). It uses base-2 exponentials for the running-max softmax, multiplying both the score scale and the bias by $\log_2 e$. Bias values are loaded directly into the register layout of the score tile, avoiding shared-memory layout conversions. Separate loops process fully causal prefix tiles and masked diagonal tiles. The launch uses 64×64 query/key tiles, four warps, and three pipeline stages. The measured specialization uses $d_h = 64$ and power-of-two lengths. Its storage avoids $n \times n$ intermediates, and its computation remains quadratic in sequence length. Dyadic pretraining uses differentiable FlexAttention.

Table 13 times the full model: attention, MLPs, offset-vector construction, and the logits head. Each of 20 complete PG-19 windows is timed with CUDA synchronization, after three untimed warmup forwards. Data preparation is excluded. On H100, two-window tests of the deep-16k checkpoint at 2048 and 8192 tokens, with and without occupancy correction, differ from FlexAttention by at most 0.0055 PG-19 perplexity (0.049%), using second-half-window scoring and an identical flat bias. We additionally benchmark the same checkpoints and kernel on an RTX 4090 under the same timing protocol (Table 14).

Table 13: Full-model prefill throughput on an NVIDIA H100 80GB HBM3; batch size 1, float32 weights with bf16 autocast, 20 complete PG-19 windows per evaluation context. The RoPE control is pretrained at 2048; the dyadic checkpoint is finetuned through 16,384. Last row: time-per-token ratio without occupancy correction.

tok/s	2048	8192	32768	65536
RoPE control (PyTorch SDPA)	99,716	97,430	68,258	47,877
deep dyadic, fused Triton	93,481	87,017	53,076	33,946
+ occ. renorm.	92,007	86,631	53,092	33,961
Deep / SDPA time per token	1.07×	1.12×	1.29×	1.41×

Occupancy correction reuses the same flat-bias lookup. Measurements exclude backward and autoregressive decoding. PyTorch selects the SDPA backend.

Finetuning compute accounting. The full recipe uses $1500/20{,}000 = 7.5\%$ of the pretraining updates and tokens. With tokens per update fixed, attention matrix-product work grows linearly with context length. An update mix of eleven 2k updates and one 16k update therefore uses

Table 14: Full-model prefill throughput on an NVIDIA RTX 4090; batch size 1, float32 weights with bf16 autocast, 20 complete PG-19 windows per evaluation context. The RoPE control is pretrained at 2048; the dyadic checkpoint is finetuned through 16,384. Last row: time-per-token ratio without occupancy correction.

tok/s	2048	8192	32768	65536
RoPE control (PyTorch SDPA)	39,848	28,407	19,515	14,331
deep dyadic, fused Triton	40,461	28,595	19,425	13,478
+ occ. renorm.	40,815	28,857	18,809	13,383
Deep / SDPA time per token	0.98×	0.99×	1.00×	1.06×

$0.075(11/12 + 8/12) = 0.11875$, or 11.875%, of the pretraining attention matrix-product work. The approximately 8% figure quoted in the main text also includes context-independent dense projections and MLP work. These estimates concern operation counts, rather than measured runtime.

Evaluation. We concatenate and tokenize PG-19 test books (4.2M tokens) and split them into non-overlapping windows of length L. Perplexity is the exponential of mean NLL over the second half of each window, scoring at most 3M tokens per context length.

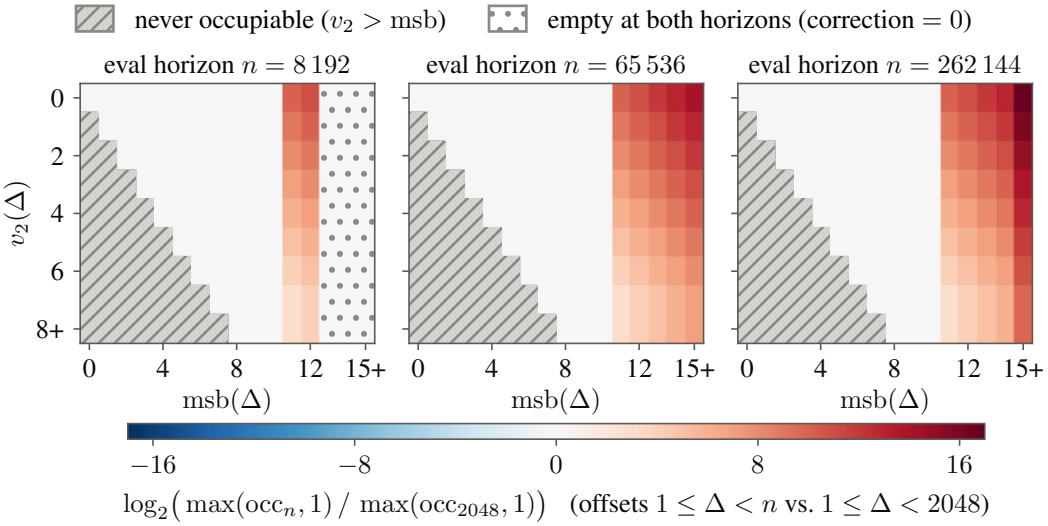

Figure 9: Log-count ratios $\log_2[\max(K_n, 1)/\max(K_{2048}, 1)]$ over the dyadic grid at three horizons. Counts use $1 \le \Delta < n$, with v_2 capped at 8 and msb at 15. The score correction is $-\lambda$ times the corresponding natural-log ratio. Dots mark cells empty at both horizons, and hatching marks structurally unreachable cells. The zero-reference convention is analyzed in Appendix P.

Correction settings. Figure 9 shows the change in cell counts. The counting argument and treatment of empty cells are given in §5 and Appendix P. All tested $\lambda \in [0.5, 1.5]$ improve the shallow model. Table 15 gives the additional strengths and the corrected tail model. Extending a table through scale 15 creates two columns beyond the 16k finetuning range, whose largest positive offset has $\mathrm{msb} = 13$.

Finetune configurations. Both configurations start from the final RoPE+dyadic checkpoint and use 1,500 updates (7.5% of pretraining), learning rate 5×10^{-5}, and 100 warmup steps. The tail variant adds one slope per head, giving bias $B[v, 8] + s_h \cdot (\mathrm{msb}(\Delta) - 8)$ for $\mathrm{msb} > 8$. The deep variant extends columns from $\mathrm{msb} \le 8$ to $\mathrm{msb} \le 15$, initializing the new columns from column 8. Tail finetuning uses context 2048 throughout. For deep-16k, every twelfth update uses context

16,384 with micro-batch size 2, and other updates use context 2048. All parameters are trainable. The deep-64k configuration instead uses 65,536-token long-context updates under the same token budget.

Table 15: Additional correction settings for the 1.3B shallow and tail models. Both use 2048-token pretraining, and the tail finetune also uses context 2048. Column headings are evaluation lengths; perplexity scores the second half of each window. The uncorrected models and standard $\lambda = 1$ shallow/deep results are in Table 10.

Configuration	2048	4096	8192	16384	32768
Shallow, $\lambda = 0.5$	17.5	56.6	293.3	355.1	400.6
Shallow, $\lambda = 1.5$	17.5	47.5	202.0	296.2	397.4
Tail, $\lambda = 1$	17.4	52.6	254.6	333.2	373.8

P CORRECTION GRANULARITY AND POSITIONAL-AXIS ABLATIONS

Global and per-query correction. The main experiments use full-window counts $K_n(c)$ shared across queries. A causal query at position i instead sees offsets $1, \ldots, i-1$. On common positive support its per-query correction is $b(c) \mapsto b(c) - \log[K_i(c)/K_{\min(i,T)}(c)]$. It vanishes for $i \le T$ and equals the global rule at $i = n$. The conditions of Proposition 1 apply at the query horizon.

On one deep-16k checkpoint, per-query perplexities at 4k/8k/16k/32k/65k are $18.16/18.27/18.13/18.68/18.85$, compared with $18.19/18.24/18.11/18.52/18.84$ for global counts. The largest difference is 0.17. Both evaluations score the second half of the same PG-19 windows, separately from the main grid. Despite this similar perplexity, correction vectors can differ: the cell $(r, s) = (0, 15)$ has count 1 at query 32,770 and 16,384 at query 65,536, a log-count difference of approximately 9.7. For the shallow model, per-query values are $55.6/287/369/410$ at 4k–32k, versus $52.6/253/328/370$ globally. Both retain poor long-context perplexity.

Boundary convention. We fix $T = 2048$ throughout pretraining and finetuning, count positive offsets only, and floor both counts at 1. A cell absent at T therefore receives $-\lambda \log K_H(c)$ when populated at H. A cell absent at both horizons receives zero. The correction is zero at $H = T$ and for the self offset. Retrieval uses the same global per-offset correction. The following ablation examines the zero-reference convention explicitly.

P.1 CELLS ABSENT AT THE REFERENCE HORIZON

For the deep map $c(\Delta) = (\min(v_2(\Delta), 8), \min(\mathrm{msb}(\Delta), 15))$, all offsets of any cell populated at $T = 2048$ are already below 2048. Their counts therefore remain unchanged at any $H \ge T$. At $H = 65{,}536$, 63 cells are common to both horizons and 45 are new. Thus correcting only common-support cells is identical to no correction, while correcting only new cells reproduces the full correction. We checked these identities with integer counts and the implemented correction tensors. Paired forwards with one backend also give identical logits. This locates the intervention outside the common-support case of Proposition 1. Some of these cells occur during finetuning: context 16k covers scale indices 11–13, while indices 14–15 lie beyond that range.

Sensitivity to the zero-reference policy. For cells with $K_T(c) = 0 < K_H(c)$, replace the reference pseudocount 1 by $\eta > 0$, keeping the positive-reference rule unchanged:

$$\delta_\eta(c) = -\lambda \log\left(\frac{K_H(c)}{\eta}\right), \qquad K_T(c) = 0 < K_H(c).$$

Table 16 compares $\eta \in \{0.5, 1, 2\}$ with no correction on one fixed deep-16k checkpoint, using $\lambda = 1$ and the same fused backend. Each length uses 20 complete PG-19 windows, with their second halves scored. The windows are identical across policies. This separate window set differs from the main grid, so all four configurations, including the uncorrected baseline, are evaluated together.

Table 16: Sensitivity to the reference pseudocount for cells with $K_T = 0$. PG-19 perplexity on shared windows for the deep-16k checkpoint. Lower is better. The standard policy remains $\eta = 1$.

Policy	16,384	65,536
No correction	19.673	22.507
$\eta = 0.5$	19.408	18.829
$\eta = 1$ (standard)	19.410	18.953
$\eta = 2$	19.429	19.070

At 65,536, the standard correction reduces mean negative log-likelihood by 0.172 per scored token and improves all 20 windows. Halving or doubling the pseudocount changes mean negative log-likelihood by less than 0.007 per token (less than 0.125 perplexity), about 4% of the correction's NLL effect. At 16,384, the corresponding changes are at most 0.001 NLL per token. This is limited sensitivity to the tested pseudocount range for one checkpoint and two context lengths.

P.2 Positional-axis ablations

Figure 10 shows the learned two-coordinate bias tables. To test whether a single distance-scale coordinate is sufficient for calibration, we pretrain an msb-only model under the same recipe. Its table has one learned scalar per distance scale and head, omitting v_2. This model matches the RoPE control in distribution (VAL 9.95 vs. 9.94) and has poor uncorrected long-context perplexity (536 at 32k), yet the training-free renormalization recovers $17.4/19.9/30.6/46.8/70.8$ across 2k–32k, a 7–11$\times$ improvement without gradient updates. At 65k, perplexity is 85.5 versus 618 without correction. The complementary v_2-only table (nine scalars per head, no msb), pretrained identically, matches the control in distribution (18.0 at 2k) and collapses at length ($71/317/454/616$ at 4–32k). Here renormalization leaves perplexity unchanged to three digits at every context. The relative frequency of offsets with $v_2(\Delta) = k$ is approximately $2^{-(k+1)}$. The counts themselves grow with n, but their leading growth is common across cells, so most of the correction cancels as a shared softmax shift. Finite horizons retain boundary effects. This common-shift behavior is consistent with the measured lack of improvement in the v_2-only model.

A reduced-budget msb-only finetune. These controls leave open whether calibration also explains the msb-only model's retrieval errors. We gave the msb-only model a reduced version of the joint dyadic+PI recipe: the same interpolation scale $1/16$ and the same every-twelfth-update 16,384-token context mix, but 800 updates at batch 64×2048 ($\approx$105M tokens, $1/15$ of the joint finetune's 1.57B), with embeddings and head frozen and an 8-bit optimizer. Held-out perplexity is nearly flat ($19.4/19.1/19.0/18.8/19.2$ at 2k–32k), the same shape as the two-axis model. The retrieval profiles differ: single-needle accuracy is $0.87/0.82/0.83/0.59$ at 2k–16k versus $0.92/0.81/0.94/0.75$ for the full-budget two-axis finetune. Across 8k and 16k, mean accuracy is 0.71 versus 0.84. Both models have low retrieval accuracy at 32k (0.04 vs. 0.00), consistent with a similar observed retrieval range in these two configurations (Appendix T). The $15\times$ budget difference prevents isolating the positional axis's effect on retrieval. The comparison is descriptive for these single checkpoints.

Q Cell geometry in sparse inference and the NoPE variant

Figure 11 shows the recorded training and validation trajectories.

Extrapolating the table. Analytically extending the table's scale recursion reproduces the clamped table: its learned long-range columns are already approximately constant across scales. Rotary content scores remain length-dependent.

Sparse inference. Since long-range mass concentrates on near-diagonal dyadic cells, we can restrict inference attention to a local window plus offsets with $\mathrm{msb} - v_2 \leq 2$: 283 keys per query at 32k context (0.9% of dense) yields ppl 21.0, constant across 4k–32k. This gives a $115\times$ reduction in attention reads at a 16% ppl premium for a model trained only with dense attention.

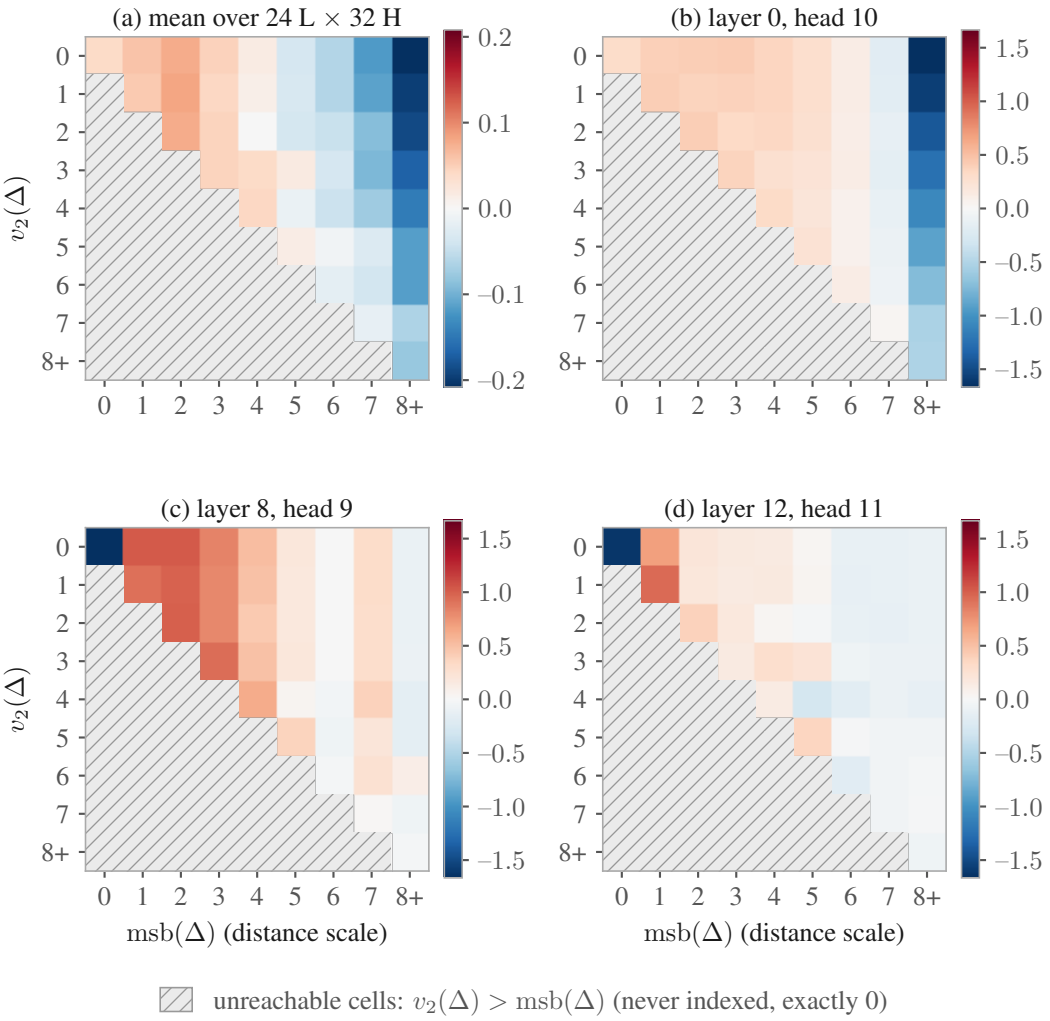

Figure 10: Learned dyadic bias tables of the 1.3B RoPE+dyadic LM at the end of pretraining (step 20,000). (a) Mean over all 24×32 heads: a recency prior over $\mathrm{msb}(\Delta)$, with a weaker long-range penalty at larger $v_2(\Delta)$. (b–d) The three heads with the largest $|\mathrm{bias}|$. Panels (b–d) share one color scale (± 1.67), while (a) has its own (± 0.21). Both axes are clipped at the table boundary: the last bin (8+) aggregates all $v_2 \geq 8$ resp. $\mathrm{msb} \geq 8$, so the saturated diagonal cell collects more than the exact powers of two. Hatched cells ($v_2(\Delta) > \mathrm{msb}(\Delta)$) are structurally unreachable and remain exactly at their zero initialization.

In-context state tracking. In-context state tracking (few-shot parity / running mod-10 sums in the prompt) is at chance for all variants at 1.3B/21B tokens. This contrasts with their improved long-context perplexity.

Evaluation at 65k. At 65,536 tokens (32× pretraining context), the RoPE control has perplexity 444 (seed 2: 696). RoPE+dyadic has 484 without correction and 398 with it (seed 2, corrected: 381). The raw/corrected pairs are 491/462 for tail and 2329/1722 for NoPE+dyadic. PI reaches 866. Table 2 reports the deep and long-context-finetuned RoPE configurations. The msb-only model reaches 85.5 with correction versus 618 raw. The sixteen-scalar scheme retains a 7× training-free gain even at 32×, although it remains well above the in-distribution perplexity. The evaluation-only RoPE rescalings in Appendix N are stronger long-context controls than those raw pretrained variants.

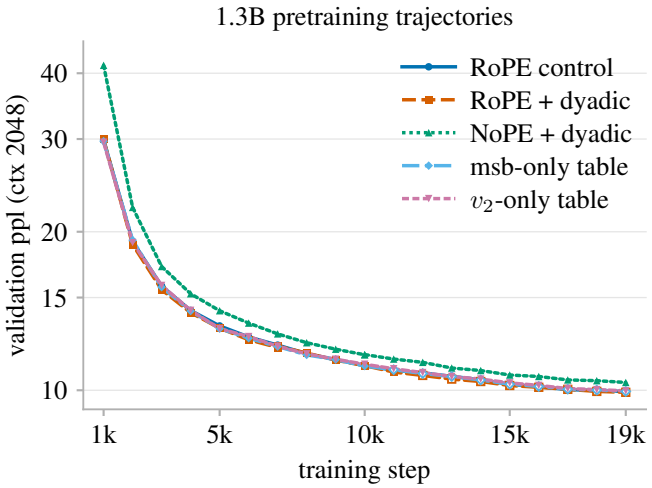

Figure 11: 1.3B pretraining trajectories (validation perplexity at the 2048-token training context, evaluated every 1,000 steps). All cell-based variants track the RoPE control throughout the measured training run. Filled markers denote recorded validation metrics. Open markers denote validation values preserved in stdout after the telemetry series ends (RoPE+dyadic from step 12k, msb-only from step 11k).

NoPE with dyadic bias. The third pretraining configuration uses the dyadic table as its only positional system. It is trained on all 8 GPUs with the same data and schedule. It converges to within 4% of the RoPE control in distribution (validation perplexity 10.35 versus 9.94). The ratio decreases from 1.39× at 1k steps to 1.04× at 19k, using only 81 bias scalars per head. On extrapolation it has the lowest perplexity of the three models at 2×–4× context (PG-19 ppl 60.5 at 4096 and 269.9 at 8192, vs. 68.3/290.2 for the control), but degrades fastest beyond 8× (759 at 16k): with the clipped window $W{=}8$, all offsets $\Delta \geq 256$ share one distance-scale bucket, so its long-range geometry is very coarse and the number of keys per bucket grows with length. Finer long-range cells would distinguish offsets merged by the clipped table.

R NATURAL-LANGUAGE EXTENSION: 19M PROBE AND THE 150M FIVE-VARIANT SWEEP

Small-scale language-model probe. We add the (unshifted, clipped) dyadic feature table as a per-head attention bias in a standard 19M-parameter causal LM trained on TinyStories (Eldan & Li, 2023) at context 512 (Table 18; second-half-window perplexity at longer contexts, three seeds with a single-seed ALiBi control). The bias matches or improves the displayed in-distribution perplexities and improves extrapolation in the tested RoPE/NoPE pairings. For NoPE, the improvement is 40× at 16× context, with smaller reported between-seed variation. ALiBi has nearly flat perplexity in this evaluation.

In the 19M implementation, CUDA rounding in the scale-index calculation maps offsets 8 and 64 to stored cells $(3, 2)$ and $(6, 5)$. These are two relocated singleton cells over the evaluated offset range. The reported perplexities use this numerical feature map.

150M control sweep. We pretrain five variants with 12 layers and width 768: RoPE control, RoPE+dyadic, ALiBi, plain NoPE, and NoPE+dyadic (Table 17). Each uses 1.57B FineWeb-Edu tokens at context 2048, with identical data order and training schedules. ALiBi has the lowest perplexity in this grid and is nearly flat beyond 8k, about 26% above its own in-distribution value. RoPE+dyadic matches the control in distribution, and correction reduces its 32k perplexity by 23% relative to the control. Correction is applied to the cell-based models; the RoPE, ALiBi, and plain NoPE evaluations are unchanged. Adding the dyadic table to NoPE reduces its 32k perplexity by $6.4\times$. These small-scale controls measure perplexity, with retrieval and matched ALiBi/plain-NoPE pretraining at 1.3B left untested.

Table 17: 150M five-variant pretraining sweep (12 layers, d=768, 1.57B FineWeb-Edu tokens each, identical data order; train ctx 2048): PG-19 perplexity, second half of window. Occ. renorm. is the training-free correction of §5.

model	2048	4096	8192	16384	32768
RoPE (control)	37.0	146.1	305.3	375.4	463.8
RoPE + dyadic	37.0	133.2	312.1	368.3	438.5
+ occ. renorm.	37.0	121.0	273.8	322.9	356.8
NoPE	42.1	142.7	789.7	4843.6	13024.5
NoPE + dyadic	40.2	95.8	569.6	1178.8	2021.1
ALiBi (recency bias)	36.4	40.0	46.1	46.2	46.4

Table 18: Small-scale LM extension (19M params, TinyStories, train ctx 512): perplexity of the second half of the evaluation window, mean±sd over 3 seeds. ALiBi has a single seed. The dyadic rows use the numerical feature indexing described in the text.

	512	1024	2048	4096	8192
RoPE	5.54±.06	11.7±0.8	39.6±1.4	77.4±5.9	106.2±11.0
RoPE+dyadic	5.53±.04	**8.8±0.5**	**23.5±1.0**	**49.1±4.6**	**74.8±6.1**
NoPE	5.96±.01	17.5±2.6	246±104	1266±991	4331±4242
NoPE+dyadic	5.82±.01	8.3±0.4	52.0±6.7	93.4±8.0	109.6±8.1
ALiBi	5.54	5.50	5.54	5.51	5.51

S SEED ROBUSTNESS AND THE 3B SCALING PAIR

Pretraining seeds. A second control/dyadic pair changes both initialization and data order. At 32k, RoPE perplexities are 480 and 692, while corrected dyadic perplexities are 350 and 386. At 16k the corresponding pairs are 401/553 and 318/310. Population mean $\pm$ standard deviation at 32k is 586 ± 106 for RoPE, 532 ± 54 for raw dyadic, and 368 ± 18 for corrected dyadic. Correction improves over RoPE at each tested length for both seeds, with varying margins (including 14% at 8k for seed 0). In-distribution values are within the observed seed spread, 17.6 ± 0.2.

Finetuning seed. A second 16k-context finetune of the same dyadic pretrain uses data-order seed 2027. Its corrected perplexities at 2k–65k are $17.7/18.2/18.2/18.1/18.5/18.8$, within 0.4 of $17.4/17.8/17.8/17.7/18.1/18.4$ for the first run. The other finetuned configurations and NoPE pretrain each have one seed.

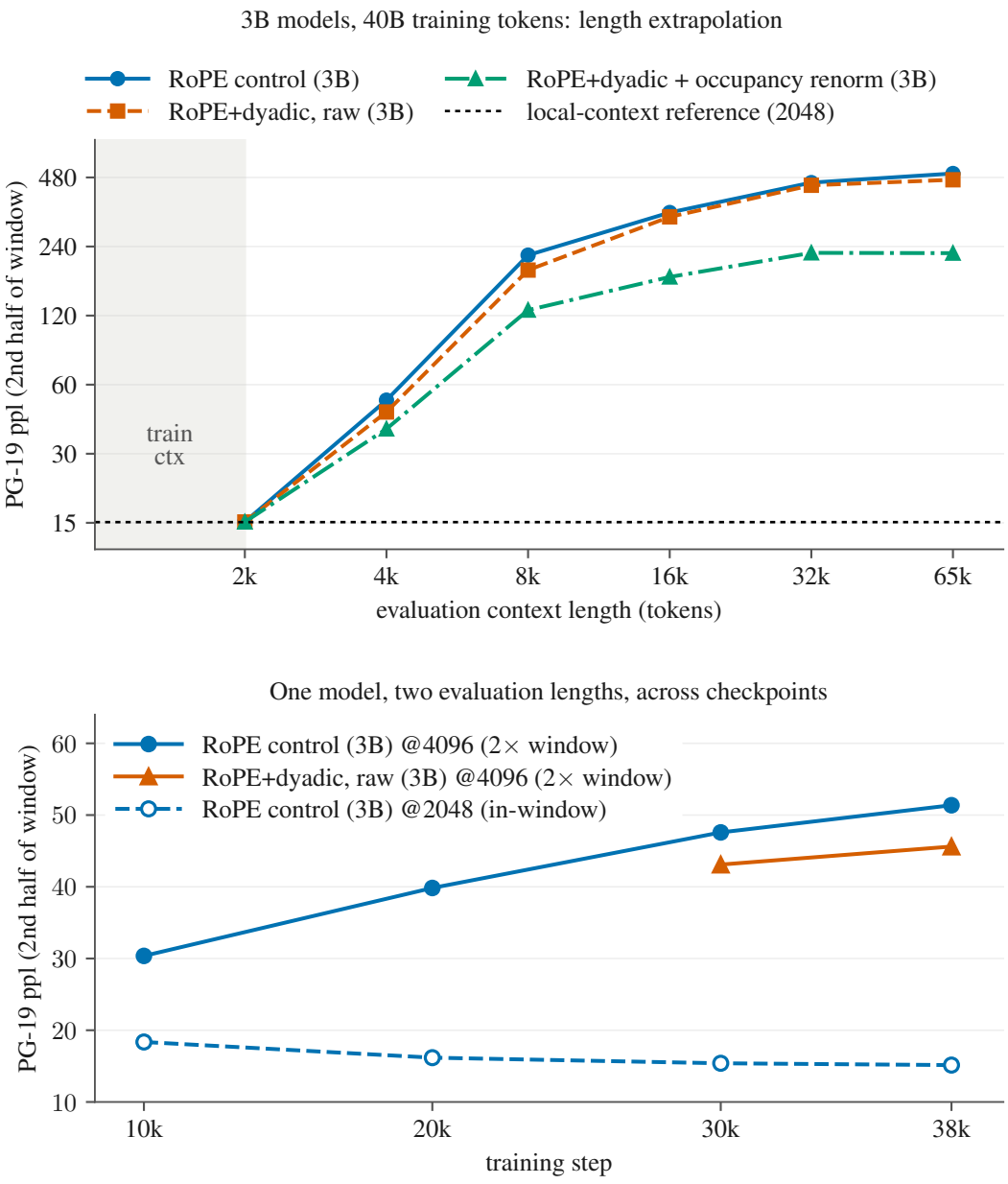

Figure 12: Top: 3B/40B-token pair. The dyadic model matches the displayed in-distribution perplexity, has lower raw perplexity beyond the training context, and the training-free renormalization widens its margin to -55% at 65k (log–log; the shaded band is the training context). The dotted black line marks the in-distribution control perplexity 15.1 as a local-context reference. Bottom: one and the same control model evaluated at its training window (2048, dashed, open markers) and at twice the window (4096, solid, filled markers) across training checkpoints.

S.1 Scaling the pair to 3B

To test whether the perplexity gains persist at larger scale, we pretrain a matched pair of 3.04B-parameter models (26 layers, d_{model}=3072, 24 heads, SwiGLU 8192; same dense architecture, tokenizer, optimizer, and schedule as the 1.3B pair) for 38,000 steps, processing 40B tokens of FineWeb-Edu, 1.9× the 1.3B token budget. Table 19 gives the full grid, and Fig. 12 shows both the length extrapolation and the control's training trajectory. This single-seed experiment evaluates pretraining and inference-time correction at 3B.

Table 19: 3B / 40B-token pretraining at context 2048, without finetuning. PG-19 perplexity uses second-half-window scoring. (a) Final checkpoints evaluated at different context lengths; relative changes compare the corrected dyadic model with the RoPE control. (b) The RoPE control evaluated at contexts 2048 and 4096 across training checkpoints. All pretraining uses context 2048.

(a) Evaluation context, final checkpoints

	2048	4096	8192	16384	32768	65536
RoPE control	15.1	51.4	220.1	337.6	455.7	499.4
RoPE+dyadic (raw)	15.1	45.6	189.5	323.1	443.7	468.9
RoPE+dyadic + occ	**15.1**	**38.5**	**126.8**	**176.8**	**225.2**	**224.6**
vs. control	0%	−25%	−42%	−48%	−51%	**−55%**

(b) Training checkpoints, RoPE control

Training updates	10,000	20,000	30,000	38,000
PPL at context 2048	18.4	16.2	15.4	15.1
PPL at context 4096	30.4	39.8	47.6	51.4

The two 3B models have the same rounded in-distribution perplexity. At longer contexts, correction reduces perplexity relative to RoPE by 55% at 65k, compared with up to 44% at 32k in the two-seed 1.3B comparison. Context and training budget differ between these comparisons, so the change cannot be attributed to parameter count alone. Within the 3B run, additional training improves in-window perplexity while worsening long-context perplexity, consistent with training-window specialization.

T Retrieval failure between 16k and 32k

Distance dependence. The joint model's drop from 0.75 at 16k to 0.00 at 32k motivates evaluations within that interval (10 trials per depth at 20,480/24,576/28,672, plus controls). The probe identifies two regimes. (a) Distance-limited addressing: at 20,480 the model retrieves needles within ∼16k tokens of the question at 0.9 while needles further away drop to 0.1–0.2 (per-depth accuracies [.1, .2, .9, .9, .9] from shallow to deep). At 24,576 the range of successful retrieval narrows ([.0, .1, .4, .7, .8]). The ∼16k radius equals the finetuning context (16,384, i.e. relative offsets ≤ 1024 in compressed units). Successful retrieval here follows the finetuned pairwise offsets rather than the nominal $\kappa \times T = 32$k interpolation range. (b) Global degradation: by 28,672 even needles 3k tokens from the question fail ([.0, .0, .1, .1, .0]). Once most pairwise offsets in the sequence are beyond the finetuned range, degradation is no longer confined to long-range pairs. Two further controls test possible explanations. Generations at 32k remain fluent and on-task ("the key that is the alpha key."). The model responds to the question but fails to retrieve the digits. Re-compressing at evaluation time (κ=32, which folds 32k into the trained span) gives 0.00 retrieval accuracy at all depths: the tested evaluation-only change in κ does not extend retrieval range. The longer-context finetune is evaluated separately in Appendix U.

U Retrieval configurations and the RULER subset

Standardized suite: a RULER subset. We evaluate four RULER tasks (Hsieh et al., 2024) using the official templates and string-containment recall (25 trials per cell, uniform needle depth): S-

NIAH-1 (essay haystack, 7-digit value), S-NIAH-2 (repeated-noise haystack), S-NIAH-3 (UUID value), and MK-NIAH-1 (three distractor needles). In Table 20, RoPE control fails beyond its pretraining context. Joint dyadic+PI has the highest mean recall across tasks at 4k/8k/16k: $0.75/0.70/0.48$, versus $0.74/0.62/0.45$ for reduced-budget YaRN-FT and $0.68/0.49/0.31$ for the paired reduced-budget PI run. It is best or tied on 8 of 12 task–context cells, while multi-key recall remains ≤ 0.40 for all models. Every 16k-finetuned configuration fails at 32k (Appendix T). The 32k-finetuned variant improves retrieval at that length. Joint dyadic+PI performs best on repeated-noise S-NIAH-2 ($1.00/0.96/1.00$ through 16k), consistent with its stronger single-key retrieval.

Table 20: RULER subset (official templates, string-containment recall, 25 trials per cell, uniform depths). All models start from 2k pretraining. The PI, YaRN, and joint rows use 16k-context finetuning; the bottom block uses 32k-context finetuning and $\kappa = 32$. The † YaRN/PI pair uses 196.608M tokens per run, versus 1.573B for the 16k joint model. The 32k joint run uses 196.608M tokens with embeddings and readout frozen.

	task	4096	8192	16384	32768
RoPE control	S-NIAH-1	.00	.00	.00	.00
	S-NIAH-2	.00	.00	.00	.00
	S-NIAH-3	.00	.00	.00	.00
	MK-NIAH-1	.00	.00	.00	.00
PI-FT†	S-NIAH-1	.80	.68	.28	.00
	S-NIAH-2	.84	.72	.56	.00
	S-NIAH-3	.72	.32	.16	.00
	MK-NIAH-1	.36	.24	.24	.00
YaRN-FT†	S-NIAH-1	.84	.68	.36	.00
	S-NIAH-2	**1.00**	**1.00**	.88	.00
	S-NIAH-3	**.80**	**.56**	**.20**	.00
	MK-NIAH-1	.32	.24	**.36**	.00
Joint dyadic+PI	S-NIAH-1	**.88**	**.92**	**.48**	.00
	S-NIAH-2	**1.00**	.96	**1.00**	.00
	S-NIAH-3	.72	**.56**	.16	.00
	MK-NIAH-1	**.40**	**.36**	.28	.00
dyadic + PI(κ=32), FT@32k	S-NIAH-1	.48	.32	.16	**.12**
	S-NIAH-2	.80	.76	.68	**.72**
	S-NIAH-3	.08	.00	.00	.00
	MK-NIAH-1	.04	.00	.12	.00

Figure 13 combines the passkey curve with multi-needle retrieval and variable tracking. The latter probes require selecting among distractors or following several assignments.

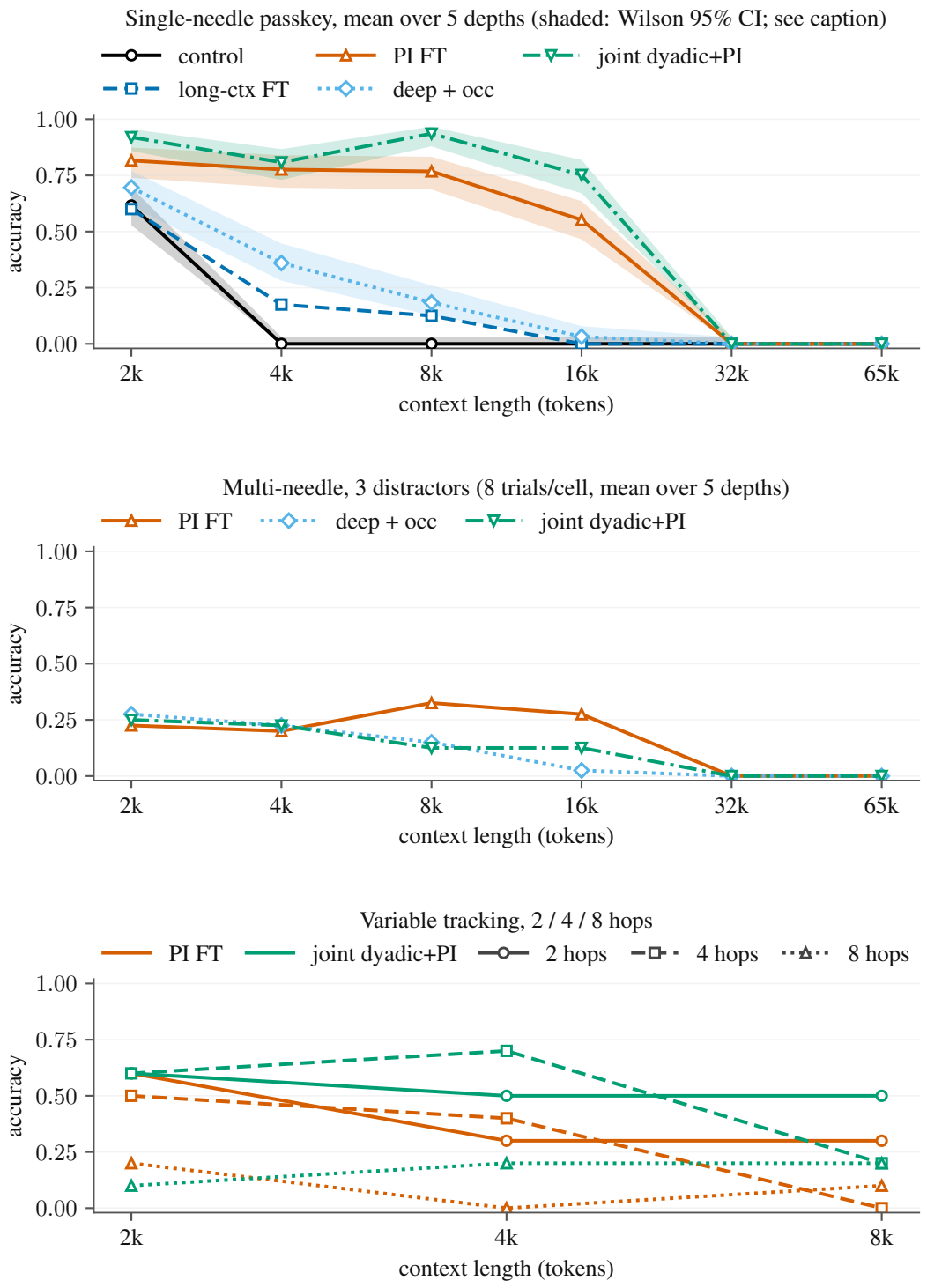

Figure 13: Task-level evaluation across configurations, stacked top to bottom: single-needle passkey means over depth (top; shaded bands are Wilson 95% confidence intervals from 25 trials per depth, n=125 per point; the long-context finetune and all 65k points use 8 trials per depth). Middle: multi-needle retrieval with three distractors, where PI and joint dyadic+PI have similar accuracy. Bottom: natural-language variable tracking at 2/4/8 hops. The joint model leads at 2–4 hops. Accuracy is low for all three configurations at 8 hops.

V EXTENDED RELATED WORK

Position and length generalization. Positional encodings strongly affect extrapolation (Kazemnejad et al., 2023; Ruoss et al., 2023; Li et al., 2024; He et al., 2024; Golovneva et al., 2024). Position Coupling and Abacus encode task structure in position IDs (Cho et al., 2024; McLeish et al., 2024). Our geometric features instead describe binary scale and divisibility. BiPE uses a two-level hierarchy, whereas the hard dyadic scan repeats the same merge across a number of levels that grows with sequence length.

Looped computation and log-depth expressivity. Universal and looped transformers reuse computation across depth (Dehghani et al., 2019; Giannou et al., 2023; Fan et al., 2025; Deng et al., 2026). Fan et al. study length generalization with $T = n$ iterations. LT2 combines looping with subquadratic attention. Our scan uses $\lceil \log_2 n \rceil$ iterations with an iteration-synchronized dyadic mask. Liu et al. (2023) give $O(\log n)$-depth non-uniform simulations of automata, and Merrill & Sabharwal (2025) establish uniform looped recognition of regular languages at $\Theta(\log n)$ repetitions. Our Theorem 1 concerns all prefix outputs and a single-block Kogge–Stone realization. Theorem 2 supplies a finite condition that can be checked for a trained network. Relevant expressivity limits are studied by Merrill et al. (2024); Hahn (2020). Log-depth recurrent units (Pert et al., 2026) obtain near-perfect OOD accuracy on regular languages through balanced binary reduction and report monoid-clustered states, which is consistent with our codebook observations.

RegularGPT and finite-state composition. RegularGPT (Chi et al., 2023) shares weights across a logarithmic number of sliding-dilated attention layers. At chunk size two, its mask reads self and the partner at offset 2^t. This is the same interior communication pattern as our binary scan. Its two-layer ReLU construction for finite-state transition composition is also a close precursor to Theorem 1.

The attention operators and deployment differ. RegularGPT adds learned relative-position scalars to content-based attention scores. DST uses three learned role logits for content-free mixing over self, partner, and an explicit sentinel. This gives two fixed mixing patterns for combine and boundary/keep updates. For projected inference, DST returns each post-loop token state to a fixed, training-derived codebook. The fixed-sentinel deployment then admits finite checks of initialization, transitions, projection decisions, and readout, with rounding-error bounds. RegularGPT analyzes transition representations and their clustering. Our verification addresses the complete specified numerical transition system of a trained model. The transfer condition also applies to non-attention updates satisfying the same hypotheses.

Hierarchical attention and binary-tree computation. H-Transformer-1D (Zhu & Soricut, 2021) and Fast Multipole Attention (Kang et al., 2023) use multiresolution hierarchies for efficient attention. LongNet (Ding et al., 2023) uses exponentially dilated attention, and PowerAttention (Chen et al., 2025) uses power-of-two offsets to grow the receptive field. SCOPE (Qi et al., 2026) assigns exponentially scaled look-back scopes to attention heads. This also provides multiscale position information, but through head-wise scope masks rather than our learned bias over (v_2, msb) cells. WAT (Berezkin, 2026) reduces sequences through a shared binary-tree GLU merge. Our evaluation of WAT finds chance-level final-answer accuracy beyond one doubling on parity and at all tested lengths on S_5 (Appendix I). Our analysis focuses on the combination of iteration-synchronized scan routing, sentinel semantics, finite-codebook verification, and positional-cell occupancy correction.

Wavelet and multiscale representations. Ruscio et al. (2024) study wavelet-like structure that emerges in RoPE attention, rather than introducing a wavelet positional encoding. Irani & Metsis (2025) propose an explicit signal-aware wavelet encoding for time-series transformers. Our hard scan uses shifted binary features, whereas the soft LM uses a clipped finite partition. These constructions have different invariance and support assumptions.

Attention sinks and parallel scan. StreamingLLM (Xiao et al., 2024) studies the role of initial tokens in streaming attention. Our sentinel has a specific computational role: it distinguishes a missing scan partner from a partner holding the query's own state (Proposition 2). Parallel scan itself is classical (Blelloch, 1990). It also enables parallel evaluation of linear recurrent networks and state-space layers (Martin & Cundy, 2018; Smith et al., 2023).

Context extension. LongRoPE and LongRoPE2 (Ding et al., 2024; Shang et al., 2025) search nonuniform rotary rescalings and combine them with finetuning. LongRoPE2 reports near-lossless 128k extension on Llama-3-8B. TAPA (Wang et al., 2025) learns a token-aware phase function inside attention, whereas our positional bias is indexed by a fixed partition of offsets. DroPE (Gelberg et al., 2025) removes RoPE after pretraining and uses a short recalibration stage. Our occupancy analysis instead characterizes additive cellwise corrections under stationarity and common-support assumptions (Proposition 1). Direct matched-budget checkpoint comparisons with these methods remain future work.

W Reproduction Guide

Algorithmic models and verification. Appendix D specifies architectures, training, evaluation, and codebook extraction. The raw-parity and S_5 checkpoints contain $199{,}410$ and $1{,}867{,}624$ trainable parameters. The fixed-sentinel certificate uses the pinned update maps, embeddings, codebook, sentinel, and decoder described in Appendix E. The empirical raw and projected evaluations follow the task definitions in Appendix A. The numerical raw-inference analyses in Appendix F have their own stated scope.

Language models. Appendices M–O specify the main grids, rescaling baselines, training schedules, and inference implementation. Appendix S describes the additional seed and 3B experiments. The deep-64k table has 18 scale columns. The pretraining tables are saved every 500 updates, and Fig. 10 uses the final 20k-step table. Evaluation fixes tokenization, window boundaries, target indices, and the correction horizon. Passkey, variable-tracking, and RULER protocols are specified in Appendices A, M, and U. Tables 13 and 14 distinguish the two hardware measurements.