DUAL-MOD ATTENTION: TOKEN RECURRENCE BEATS LOOPING

Anonymous authors
Paper under double-blind review

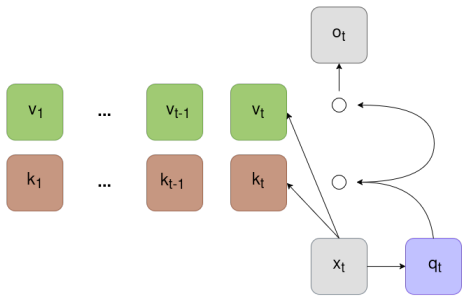
ABSTRACT

Transformers have shown incredible gains in reasoning performance by scaling parameter count and test time compute, but together they create an immense increase in computation budget and context lengths. Latent-thinking approaches grow computation without emitting tokens, along two paths: looped transformers, which iterate a weight-tied block, and recurrent hybrids, which carry a compact evolving state. Both of these methods have their own failure modes. Looped transformers can solve some problems outside of TC^0, but their per-iteration update shrinks and accuracy saturates, after which further iterations add nothing. Their reach is capped by the training-time iteration count and length, and we find that extra test-time iterations do not extend it. Past the training length, looping rapidly collapses to chance and stays within a transformer's length-generalization boundary. Recurrent methods are better at length generalization and also solve tasks outside of TC^0, but carry a fixed $\Theta(d)$ state that provably bounds them as the number of tracked states grows. Linear recurrences commonly used in hybrids cannot perform NC^1-complete state tracking: diagonal ones remain in TC^0 outright (Illusion of State), and single-step delta-rule ones confine their transition spectra to [0,1]. We present a new path to scaling computation using a non-linear recurrence within softmax attention. Our method uses an append-only KV-tape where the entries are computed recurrently from the tape instead of raw tokens. We call this class of architectures token recurrences. Our arch is an exact superset of transformers whose chunk sweeps have a unique fixed point, the sequential recurrence itself, reached in finitely many steps, and whose state moves with every token. Since AR decoding is already sequential, the recurrence adds no decoding cost. Empirically, a single layer of our arch solves instances of the P-complete circuit value problem up to depth 1024. It also solves the NC^1-complete A_5 word problem, and on a variant with K interleaved A_5 registers tracks up to 1024 registers at full depth, where single-pass attention and fixed-state baselines stop by 32 and 128 registers. Token recurrences cross the boundary that loops cannot while keeping a growing cache, and ours allows chunk-parallel training at a fraction of a layer per sweep. On language modeling tasks, it beats parameter matched baselines on loss and is competitive with SOTA arches in each eval suite.

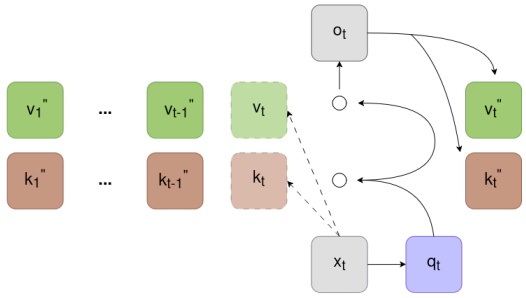
1 INTRODUCTION

Transformer based language models have shown incredible reasoning capabilities that have allowed them to solve complex math, science and agentic coding problems. This capability has largely come from scaling parameter count and test time compute (Snell et al., 2025) through emitting chain-of-thought (Li et al., 2024) tokens. In a standard transformer the softmax attention operation writes a key and value to an append-only tape, then performs a weighted soft lookup over the values of preceding tokens. The computation performed by such an operation is shallow. At fixed depth and precision a transformer is in TC^0 (Merrill and Sabharwal, 2022; Hao et al., 2022; Hahn, 2019), so problems requiring sequential composition of the prefix are out of reach in a single pass, no matter the parameter count. Chain-of-thought recovers sequential computation, but every serial step emits a token that all subsequent steps must attend over, so both memory and compute grow with the length of the reasoning chain.

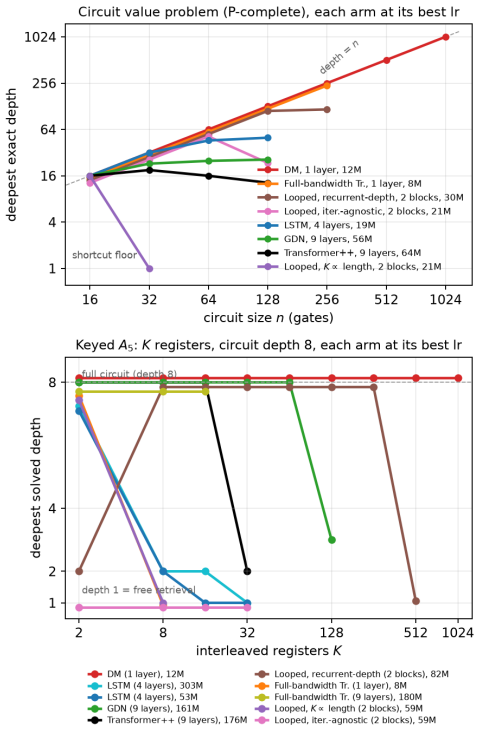
Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping 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.



(a) Standard self-attention: write, then read.



(b) Dual-Mod attention: read, then write.



(c) Serial depth (top) and growing state (bottom).

Figure 1: **Dual-Mod attention.** (a) Self-attention commits $\mathbf{k}_t, \mathbf{v}_t$ to the KV-tape and then attends. (b) Dual-Mod first attends over previous entries and the current raw pair (dashed), then writes a modded pair $\mathbf{k}_t'', \mathbf{v}_t''$ (Eqs. 2–4). (c) Deepest circuit depth evaluated exactly against circuit size on CVP (top) and deepest composition depth solved against the number of interleaved registers on keyed A_5 (bottom), every arm at its best learning rate. A one-layer DM tracks depth $= n$ to 1024 and composes fully to K=1024. Numbers are in Table 3.

Recent methods instead scale computation in latent space, without emitting tokens, by looping layers or maintaining a recurrent state (Saunshi et al., 2025). Looping allows an adaptive increase in computation per token and can leave TC^0. Recurrent methods carry a fixed $\Theta(d)$ state that can leave TC^0 based on the update rule. Hybrids of linear recurrence and softmax attention now run at frontier scale (Qwen Team, 2026; Bai et al., 2026).

In practice both families have characteristic failure modes. Looped transformers converge to fixed-point attractors (Bai et al., 2019; Zhang et al., 2025; Nanbeige Lab et al., 2026). We measure this directly for three looping recipes. In every run the per-iteration update shrinks, as $1/k$ under two recipes and to a constant floor under the third, and accuracy stops changing after $k \approx 32$, at the answer where the loop forms and at a shortcut where it does not. A loop's reach is therefore capped by the iteration count and sequence length it was trained at. On the circuit value problem (Ladner, 1975) the loops wall at their iteration count, and on the A_5 word problem (Barrington, 1986) they fall to chance by 2–2.5$\times$ the training length (Fig. 2), at K forward passes per token. On regular languages outside C-RASP all three loop recipes land on the transformer's numbers (App. E). Recurrent methods have a fixed-size state that bounds the number of registers they can track concurrently, and the linear recurrences used in hybrids cannot perform NC^1-complete state tracking under the standard $\mathsf{TC}^0 \neq \mathsf{NC}^1$ conjecture: diagonal ones remain in TC^0 outright (Merrill et al., 2024), and single-step delta-rule ones confine their transition spectra to $[0, 1]$ (Grazzi et al., 2025).

We present an alternative class of architectures, token recurrences, which keep a transformer's per-token KV-tape but compute its entries recurrently from earlier positions rather than from the current token alone. Our architecture scales serial computation without growing the context by integrating

a non-linear recurrence into the write operation of the append-only tape the transformer already maintains. We name it dual-mod attention (DM) as we modify the write process of both the key and value tapes, and show that each mod serves a different purpose and their gains combine (§D.4). Instead of writing keys and values from the input hidden state and then attending, DM first attends and then writes keys and values conditioned on the attention output. This creates a recurrence along the causal tape. The computation available to position t has a dependency chain of length t, so serial depth grows with position without added layers. Additionally, each entry is a composed value written with its own key, so later positions can retrieve and update it by content. The chunk sweeps of §3.2 have a unique fixed point, the sequential recurrence itself, reached in finitely many steps, and the state moves with every token. Since AR decoding is already sequential, the recurrence adds no decoding cost. The training is parallelized in chunks (§3.2).

Empirically, a single layer of DM with 12M parameters solves instances of the P-complete circuit value problem (Ladner, 1975) exactly to depth 1024. A parameter-matched LSTM walls at depth 48–64 and GDN at 24–28, the transformer is bounded by its layer count and looped transformers by their iteration count. DM also solves the NC^1-complete A_5 word problem (Barrington, 1986) at length 64 with one layer, where the transformer needs nine layers and does not extrapolate, and where gated linear recurrence walls by position 16. On a keyed variant with K interleaved registers, each composed to depth 8, DM tracks 1024 registers. The transformer drops out by 32 registers, the LSTM by 8 and GDN by 128. Every other architecture we test walls on at least one of these two axes or pays K passes per token to reach them. Token-recurrent models length-generalize outside C-RASP where transformers fail past the training range, DM's formed parity seeds to 8–10$\times$ (App. E). On language modeling at 340M and 1.3B parameters, DM beats its parameter-matched controls on loss and is competitive with SOTA hybrids at matched scale across evaluation suites.

In summary our contributions are (1) a new attention variant that is an exact superset of self-attention, (2) empirical evidence that it solves state-tracking problems beyond the reach of the tested transformers and fixed-state recurrences, and length-generalizes beyond standard transformers on languages outside C-RASP, (3) a direct measurement of iterate settling in three looped-transformer recipes, showing that a loop's reach is capped by its trained iteration count and length, while the recurrence advances one step per token at no extra decoding cost, (4) competitive language modeling at 340M and 1.3B against parameter-matched and state-of-the-art baselines, and (5) an efficient training pipeline that trains the model in a spectrum between fully parallel and fully sequential.

2 Related Work

DM can be compared against three lines of work. The first tries to prevent loop collapse by treating the fixed point as something to avoid, through input injection and progressive loss (Bansal et al., 2022), step embeddings (Dehghani et al., 2019), length-tied iteration counts (Fan et al., 2025), randomized initial states (Anil et al., 2022), noise-initialized state with randomized iteration counts and truncated backpropagation (Geiping et al., 2025), and memory highways (Yu et al., 2026). We implement all but progressive loss and memory highways and find that under every one of them the update shrinks and accuracy saturates (§5). Deployed looped language models accordingly report convergence after a few loops (Nanbeige Lab et al., 2026), since a token stream does not say how many iterations an instance needs.

The second line places a fixed state beside the growing KV tape (Cui, 2026; Lufkin et al., 2026; Tan et al., 2026; Li et al., 2026; Du et al., 2026). PaTH (Yang et al., 2025) and DeltaFormer (Xu et al., 2025) instead add expressivity inside attention through a compositional transform that stays fixed in size, and non-diagonal linear recurrences such as DeltaProduct (Siems et al., 2025) escape the single-step delta-rule spectrum bound. In all of these, what composes is a fixed-size object. In DM the composed state is the tape entry itself.

The third line, and the closest to ours, is the full-bandwidth transformer (FBT) (Wang et al., 2026), a token recurrence descended from the Feedback Transformer (Fan et al., 2021) that feeds each position's final hidden state into the next input so the whole stack recurs over the token stream; T^2MLR (Cai et al., 2026) loops the same carry from a middle layer to an earlier one. Trained exactly, FBT tracks serial depth, holds keyed state and length-generalizes as far as the LSTM. The difference is where the recurrence enters. FBT composes in one d-dimensional vector carried from the adjacent position, so its exact recurrence trains only sequentially in T, and under DM's chunk

scheme every sweep recomputes the stack, $25\times$ a parallel pass at the 1.3B schedule against DM's $3.3\times$. The few-pass approximation Wang et al. (2026) scale instead trains for stability under one to three passes, forms none of our tasks at its own schedule or at DM's cost, and gains reach only with the pass count, as a loop does (App. F). DM composes at the attention read, from whichever entry the query selects, and writes the result back under its own key, at a fraction of a layer per sweep.

3 Dual-Mod Attention

3.1 Method

Dual-Mod (DM) attention inverts the order of operations in self-attention (Vaswani et al., 2017): it reads first and uses the result to compute the entry it writes (Fig. 1b), which creates a recurrence along the tape.

In both cases the layer generates $\boldsymbol{q}_t$, $\boldsymbol{k}_t$ and $\boldsymbol{v}_t$ from its input $\boldsymbol{x}_t$, and the causal read is

$$\mathrm{Attn}(\boldsymbol{q}_t;\ \{\boldsymbol{k}_i, \boldsymbol{v}_i\}_{i\le t}) = \sum_{i\le t} a_{ti}\, \boldsymbol{v}_i, \qquad a_{ti} = \mathrm{softmax}_i\Big(\boldsymbol{q}_t^{\mathsf{T}} \boldsymbol{k}_i / \sqrt{d}\Big)\,. \tag{1}$$

Vanilla attention commits $\boldsymbol{k}_t, \boldsymbol{v}_t$ to the tape and computes $\boldsymbol{o}_t = \mathrm{Attn}(\boldsymbol{q}_t;\ \{\boldsymbol{k}_i, \boldsymbol{v}_i\}_{i\le t})$ (Fig. 1a). Dual-Mod does not commit $\boldsymbol{k}_t, \boldsymbol{v}_t$. It reads over the previously written entries together with the current raw pair, and the read output determines the modded pair $\boldsymbol{k}''_t, \boldsymbol{v}''_t$ that is appended (Fig. 1b):

$$\boldsymbol{o}_t = \mathrm{Attn}(\boldsymbol{q}_t;\ \{\boldsymbol{k}''_{<t}, \boldsymbol{v}''_{<t}\} \cup \{\boldsymbol{k}_t, \boldsymbol{v}_t\}) \qquad \boldsymbol{u}_t = \mathrm{RMSNorm}(\boldsymbol{o}_t) \tag{2}$$

$$\boldsymbol{k}^{\mathrm{ctx}}_t = \lambda\, \mathrm{RMSNorm}\big(\boldsymbol{W}_k\, \boldsymbol{u}_t\big) \qquad \boldsymbol{v}^{\mathrm{ctx}}_t = \boldsymbol{W}_v\big(\boldsymbol{u}_t + \sigma(\boldsymbol{G}\boldsymbol{x}_t) \odot \tanh(\boldsymbol{H}\boldsymbol{u}_t)\big) \tag{3}$$

$$\boldsymbol{k}''_t = \boldsymbol{g}^k_t \odot \boldsymbol{k}_t + (1 - \boldsymbol{g}^k_t) \odot \boldsymbol{k}^{\mathrm{ctx}}_t \qquad \boldsymbol{v}''_t = \boldsymbol{g}^v_t \odot \boldsymbol{v}_t + (1 - \boldsymbol{g}^v_t) \odot \boldsymbol{v}^{\mathrm{ctx}}_t \tag{4}$$

with per-dimension gates $\boldsymbol{g}^p_t = \sigma(\boldsymbol{W}^p_g[\boldsymbol{x}_t; \boldsymbol{u}_t])$ for $p \in \{k, v\}$, per-head RMSNorm in the $\boldsymbol{k}$ write and a learnable gain λ. Blending happens before RoPE, the cache stays append-only with $\boldsymbol{k}''_t, \boldsymbol{v}''_t$ stored in place of $\boldsymbol{k}_t, \boldsymbol{v}_t$, and the output path is that of Vaswani et al. (2017) ($\boldsymbol{W}_o\boldsymbol{o}_t$ + residual).

The modded pair at t depends only on entries before t and the raw pair at t, so the recurrence in Eq. 2–4 is strictly causal and one left-to-right pass computes it exactly. A parallel sweep over a chunk of c positions makes position $r{+}1$ correct after $r{+}1$ sweeps, so the sweep map reaches that computation in at most c sweeps whether or not it is contractive, and has no other fixed point to settle into. Autoregressive decoding is its exact solver, and training uses the chunked scheme of §3.2.

The gated blend makes Dual-Mod an exact superset of self-attention. At $\boldsymbol{g} \equiv 1$ the contextual branch disappears and $\boldsymbol{k}''_t = \boldsymbol{k}_t$, $\boldsymbol{v}''_t = \boldsymbol{v}_t$. At $\boldsymbol{g} \to 0$ the layer is pure recurrence. We initialize the gates at 0.5 so that both branches start with equal mixture weights; a near-vanilla initialization degrades performance (App. D.4).

The $\boldsymbol{v}$ write is inspired by the LSTM update with an added residual path, which we call *mult-res*. The bounded $\tanh$ branch is the composition path, the identity path carries the payload, and the boundedness stabilizes training. The $\boldsymbol{k}$ write is a normalized linear projection with no learned activation, and adding one gave no benefit (App. D.3).

We use rank-$\lceil d/8 \rceil$ or $\lceil d/16 \rceil$ factorizations for $\boldsymbol{W}^k_g$, $\boldsymbol{W}^v_g$, $\boldsymbol{H}$, $\boldsymbol{G}$, $\boldsymbol{W}_k$ and $\boldsymbol{W}_v$, which is a better allocation of the parameter budget than full rank (App. D.2). Because the modded pair replaces the raw pair in the cache, KV memory equals a standard transformer's, and because $\boldsymbol{o}_t$ is reused for both the layer output and the modded write, each layer performs one attention operation, unlike looped models (Dehghani et al., 2019).

3.2 Chunk-Parallel Training

The recurrence in DM has no associative scan, so computing the tape for a whole sequence in parallel takes several passes, which raises training FLOPs over a parallel model. Since the recurrence is causal, the passes can be scheduled so that training cost lies on a controllable curve between fully sequential and fully parallel, while AR decoding costs nothing extra.

Our engine (WaveScan) runs a chunked scheme in a fused kernel. The sequence is split into chunks of c tokens. Within a chunk the tape is initialized from the raw pairs and K parallel sweeps update

all positions at once, each sweep reading the previous sweep's entries. Committed chunks are frozen and never revisited (Jacobi iters intra-chunk and Gauss–Seidel across chunks; App. C.4 states what freezing means for the gradient). Since position j reads only positions $< j$, position $r{+}1$ is correct after $r{+}1$ sweeps, so $K{=}c$ sweeps reproduce the exact sequential computation regardless of contraction. Sweep 0 is a standard transformer, and the (c, K) family interpolates from vanilla attention to the exact recurrence.

c and K set different costs. Serial depth is $(T/c) \cdot K$, with each sweep a parallel pass over c tokens, rather than T single-token steps. FLOPs are about $(1 + Kf)$ times a parallel pass, where f is the fraction of a layer recomputed per sweep (this includes the attention reads and contextual writes, while the trunk is computed only once) and does not depend on c. A sweep has to recompute whatever the recurrent state flows through, and in DM that is the attention read and a low-rank write, the smallest part of a layer that reads the tape, so $f \approx 0.1$. Within a chunk, K bounds the composition depth, since each sweep adds one relay step on top of attention's unbounded depth-1 lookup. Exactness needs $K{=}c$, but for language modeling we see that far fewer sweeps suffice. The correctness front of the learned solution advances $s \approx 16$ positions per sweep, so the sweeps required grow as $K(c) \approx a + c/s$, which puts training a 1.3B model at about $3.3\times$ the FLOPs of the parameter-matched Transformer.

4 RESULTS

We run language modeling and state tracking on formal languages. Language models are trained at 340M and 1.3B parameters. The 340M class uses exactly the setup of Tumma et al. (2026) (SlimPajama, Mistral-32k tokenizer, 15B tokens in 30k steps of 0.5M tokens, 2K context), so language modeling and commonsense results compare directly to Tumma et al. (2026) and Cui (2026). Retrieval is reported under each source's protocol, either Arora et al. (2024a) or Arora et al. (2024b). At this scale we also train Transformer++ and gated DeltaNet (GDN) controls in the same non-embedding parameter band (303.6–334.0M) on the same token stream.

The 1.3B class uses the setup of Hatamizadeh et al. (2026) (FineWeb-Edu, Llama-2 tokenizer, 100B tokens in 190k steps of 0.5M tokens, 4K context). We train two DM variants, one with full 4K attention and one with a 2K sliding window matching the GDN-2 hybrid convention, and score DM through its training operator ($K{=}24$ sweeps; the full-attention checkpoint rescored at $K{=}c{=}64$, where the chunk recurrence is exact, agrees on cross-entropy, every commonsense task and four of six retrieval tasks within a point, App. C.4). All comparisons are parameter-matched rather than FLOP-matched (§3.2). The 340M runs use $8\times$ H100 and the 1.3B runs $32\times$ B300 GPUs.

The state-tracking experiments follow the tagging protocol of Merrill et al. (2024). The input contains only operations, the target at every position is the running state, and instances are freshly sampled from the generator rather than drawn from a fixed set, with evaluation sampled independently. We compare against width-matched Transformer, LSTM, GDN, full-bandwidth transformer (Wang et al., 2026) and looped-transformer baselines, the last under the iteration-agnostic, instance-tied and recurrent-depth (Geiping et al., 2025) recipes. The main text reports A_5, keyed A_5 and the circuit value problem (CVP). Length generalization on $\mathbb{Z}_2$ parity and $\mathbb{Z}_{10}$ is in App. E. Task definitions are in App. B; model sizes are given with each task. Table 4 summarizes what each family provides.

4.1 LANGUAGE MODELING

At 340M, DM has the lowest validation loss of the three arms trained on our token stream and the best commonsense composites of every architecture reported under this setup (Table 6). Its Avg_6 of 43.65 is above PGDN and PKDA (Tumma et al., 2026), HOLA (Cui, 2026) and NHA (Du et al., 2026), and its Avg_8 of 50.12 is above every row of Tumma et al. (2026). Its margins over the parameter-matched controls are largest on recall (Tables 7 and 8). Under the Based protocol DM averages 40.31 on retrieval against 26.77 for Transformer++ and 25.06 for GDN, and under the JRT protocol 36.47 against 32.89 and 25.87, where it trails only NHA and GSA-H. On zero-shot MQAR, DM averages 58.25 across 8 to 64 pairs against 34.64 for GDN and 22.46 for Transformer++, and 1500 steps of finetuning drive all three to 1.0, so zero-shot is the only discriminating setting.

On RULER at the 2K training length, DM leads on the harder tasks, with 84.6 on S-NIAH-3, 75.2 on MK-NIAH-1 and 14.8 on MK-NIAH-2 against 64.4, 53.4 and 7.8 for Transformer++. Like Vaswani

	LM ppl ↓		Commonsense Reasoning ↑											In-context Retrieval (JRT, 2K) ↑						
Model	Wiki	LMB	LMBa	PIQA	HellaS	WinoG	ARC_e	ARC_c	Avg_6	OBQA	SIQA	BoolQ	Avg_9	FDA	SWDE	SQuAD	TQA	NQ	DROP	Avg_6
Hatamizadeh et al. (2026)																				
GDN-2 recurrent	15.90	11.41	48.09	72.80	56.84	57.85	72.43	38.23	57.71	31.60	41.59	59.54	53.11	19.98	23.65	36.75	61.37	19.64	17.87	29.88
GDN-2 hybrid (+2K SWA)	**15.62**	**10.43**	**50.90**	72.20	58.46	**58.56**	71.89	36.69	58.12	**33.00**	41.50	**62.57**	**53.97**	54.68	41.96	44.70	62.38	26.31	**23.63**	42.28
Transformer (+2K SWA)	19.22	13.72	48.32	70.21	56.12	55.85	69.23	33.84	55.60	25.00	39.74	59.42	50.86	54.78	32.21	38.67	58.09	22.49	22.18	38.07
Mamba-3 MIMO hybrid	15.81	10.92	49.82	71.98	58.19	57.06	70.54	**38.48**	57.68	29.40	40.99	57.98	52.72	55.31	32.33	44.70	59.00	26.26	23.08	40.11
KDA hybrid	16.01	10.66	49.21	71.06	56.69	57.77	71.59	35.07	56.90	30.00	41.20	62.03	52.68	53.59	39.83	40.10	59.89	25.27	22.18	40.14
Ours																				
DM (+2K SWA)	16.14	10.61	50.32	**73.07**	**59.59**	58.17	**72.69**	36.52	**58.39**	31.20	**41.61**	54.43	53.07	**63.25**	**49.41**	**45.84**	**65.23**	**27.21**	22.90	**45.64**
DM (full 4K attention)†	15.96	9.94	51.99	72.85	59.26	58.56	72.10	38.14	58.82	31.20	42.12	61.99	54.25	70.24	58.24	49.16	66.47	30.16	23.77	49.67

Table 1: 1.3B parameters, 100B tokens, every row on FineWeb-Edu with the Llama-2 tokenizer under the GDN-2 recipe (4K context, 2K SWA for hybrids) (Hatamizadeh et al., 2026). Avg_6 averages six commonsense tasks, Avg_9 adds OBQA, SIQA and BoolQ, and retrieval uses the JRT harness at 2K (Arora et al., 2024b). DM (+2K SWA) is the architecture-matched counterpart of the hybrid rows. †Full 4K attention with no window, shown for reference and excluded from marking; retrieval prompts are $\leq$2K, within the window, so its retrieval edge is not attention reach. Published 1.3B hybrids on other corpora are in App. C.2.

et al. (2017) attention it does not extrapolate past its training length, where GDN's fixed state does. In the ablations (App. D.4) k-mod alone beats v-mod alone on language modeling, even though the value path carries the composition.

At 1.3B under the GDN-2 recipe (Table 1), DM with a 2K sliding window has the best commonsense Avg_6 of every architecture trained under that recipe, 58.39 against 58.12 for the GDN-2 hybrid, and the best retrieval average, 45.64 against 42.28, leading on five of the six retrieval tasks. Its Avg_9 trails the GDN-2 hybrid on BoolQ alone. The full-attention variant, reported for reference outside the hybrids' efficiency class, is stronger on every retrieval task and on Avg_9. Published hybrids at 1.3B trained on SlimPajama with a different tokenizer are collected in App. C.2.

The sliding-window result is particularly consequential. On RULER (App. C.3) the windowed DM leads MK-NIAH-1 at 2K and, at 8K, four times its window, retains 22 to 24 on the single-needle tasks where the windowed transformer trained under the same recipe drops to zero and GDN-2 hybrid scores 27 to 29. The full-attention DM, like a plain transformer, drops to zero past its training length. A sliding window in standard attention moves information out of the window only through depth, one window per layer, while DM's recurrence carries it forward along the tape, so a windowed DM behaves as a recurrent model whose working memory is an attention window.

4.2 Expressivity ladder

We test DM on formal language tasks in increasing order of what a solution must carry. The A_5 word problem loads the circuit class, keyed A_5 loads the circuit class together with the number of tracked registers, and CVP loads a circuit depth that grows with the input. Every result below is on the tagging protocol of §4, so chance on A_5 is $1/60$.

A_5 word problem. Table 2 and Fig. 2 compare width-matched arms (d=768, looped models d=1280) trained at T=64 for 12k steps, each at its best learning rate. GDN walls by position $\sim$16 and is at chance in the deepest bin *in-distribution*. The nine-layer Transformer forms in 5/8 seeds and is otherwise at chance. The three non-linear recurrences, DM, the LSTM and the full-bandwidth transformer, are flat at 1.0 at every position. DM reaches this state end-to-end from a cold start in 25–50% of seeds at every width to 425M, and the one-layer 12M model forms in 2/2 seeds. On length extrapolation DM sits between the Transformer and the sequential recurrences. Trained at T=64, the formed DM seeds hold 1.0 in-distribution and decay to 0.95–1.00 at 2× and 0.56–0.89 at 4× (Fig. 2). The formed Transformer decays to 0.60 and 0.31, and the LSTM and the one-layer full-bandwidth transformer stay at 1.0.

Following the length-versus-minimum-depth protocol of Merrill et al. (2024) we also sweep layers at fixed width d=768 (Table 2, right; Fig. 6). The Transformer's minimum depth to reach 0.95 grows with length, 2, 3, 6, 4 and 9 layers at T=5, 10, 15, 20 and 64. DM solves every cell including T=64 with one layer (45/48 seeds formed). The one-layer DM is also the most reliable former at T=64, with 2/2 seeds against 1/2 at 2 and 9 layers and 0/2 at 3 and 5, so depth does not carry the computation and hinders cold formation. The LSTM and the full-bandwidth transformer solve every

Model	params	in-dist. (T=64) all pos.	in-dist. (T=64) deepest	extrap. 2×	extrap. 4×
Transformer++ (9L)	64M	0.995	0.976	0.597	0.308
GDN (9L)	56M	0.145	0.013	0.084	0.049
LSTM (4L)	19M	**1.000**	**1.000**	**1.000**	**1.000**
Looped (iter.-agn.)	59M	**1.000**[§]	**1.000**	0.793	0.404
Looped ($K \propto$ len.)	59M	**1.000**[§]	**1.000**	0.604	0.308
Looped (recurrent-depth)	82M	**1.000**	**1.000**	0.681	0.352
Full-bandwidth Tr. (1L)	8M	**1.000**	**1.000**	**1.000**	**1.000**
DM (1L)	12M	**1.000**	**1.000**	0.95–1.00	0.56–0.86
DM (9L)	106M	**1.000**	**1.000**	0.96–0.99	0.65–0.89

Model	min. layers at length T 5	10	15	20	64
Transformer++	2	3	6	4	9
GDN	>9	>9	>9	>9	>9
LSTM	1	1	1	1	1
Looped (iter.-agn.)	1	1	1	1	2[§]
Looped ($K \propto$ len.)	1	1	2	1	2[§]
Looped (recurrent-depth)	1	1	2	1	1
Full-bandwidth Tr.	1	1	1	1	1
DM	**1**	**1**	**1**	**1**	**1**

Table 2: A_5 word problem, tagging protocol, all arms at d=768 (looped models d=1280) and their best learning rate. **Left:** trained at T=64 and evaluated at 64/128/256. "Deepest" is positions 57–64. DM and Transformer++ rows are formed seeds (DM 2/2 at one layer, 1/2 at nine, Transformer++ 5/8). **Right:** shallowest depth at which any seed reached 0.95 at 12k steps. For looped models depth is the number of unique blocks, and "$>n$" means no depth up to n reached 0.95. Iteration-agnostic loops train at $K \sim U[K/4, K]$, "$K \propto$ length" ties K to the sequence length, "recurrent-depth" is the recipe of Geiping et al. (2025); looped rows at T=64 are formed seeds at 10^{-4}. [§]24k steps (0.995 and 0.97 at 12k).

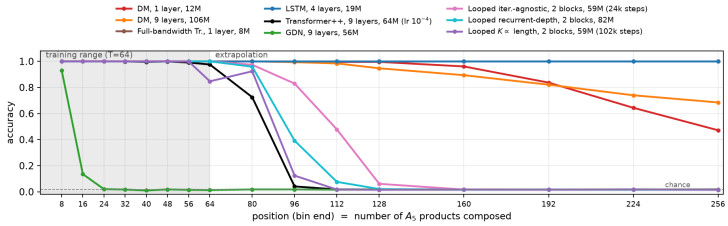


Figure 2: A_5 accuracy by position, trained at T=64 (shaded) and evaluated to 256, every arm at its best learning rate. Best formed seed shown for DM and Transformer++. The full-bandwidth transformer's curve coincides with the LSTM's at 1.0 and is hidden beneath it.

length at one layer, and GDN forms at no depth to nine at any length, its per-position reach being two products regardless of depth. The looped transformers form at one or two blocks at every length, where $K \geq T$ iterations supply the depth (§5).

Keyed A_5: capacity × depth. Tracking one A_5 register is a $\Theta(1)$-state problem, so Table 2 is a tie between the LSTM and the token recurrences. We create a keyed variant that interleaves K independent A_5 running products in one stream. Each position is tagged with its register and queries the depth-d composition of that register, so a solution must hold $\Theta(K)$ group elements and compose each to depth d. This loads the state and circuit axes together. Depth 1 is free retrieval and is excluded. We call a register depth solved at accuracy ≥ 0.95 and summarize an architecture by its state budget, K times the deepest solved depth. Table 3 (left) holds d=8 and grows K to 1024, every arm at its best learning rate and GDN under the same register-ramp recipe as DM.

A one-layer 12M DM forms cold at K=8 (2/2 seeds) and, grown by a register ramp, holds depth 8 at every K to 1024, a budget of 8192, with all six K=1024 lineages at 0.995–0.998 overall and every depth above 0.95. The LSTM's budget grows sub-linearly with its hidden size, 16 at 53M and 16–32 at 303M, and at 538M it fails depth-1 retrieval at K=16. The Transformer (9 layers, 176M) holds depth 8 to K=16 and drops to 2 at 32, a budget of 128. GDN (9 layers, 161M) holds depth 8 to K=64 and drops to 3 at 128, a budget of 512 that shrinks with its depth, consistent with a bound set by its matrix state. The iteration-agnostic loop stops at depth 1, and the instance-tied loop at depth 1 beyond K=2. The recurrent-depth loop holds depth 8 to K=256 with two blocks, at K passes per token, and none with one block at 23M or 62M. The full-bandwidth transformer holds depth 8 at K=2, 8 and 16 with nine layers and 180M parameters, its K=32 chain unfinished at the time of writing, and at K=2 only with one layer at both 8M and 23M (depth 1 at K=8 in two seeds, learning rates and width), so its addressable state comes from depth where DM's comes from the tape.

Model	params	deepest solved depth at D=8 K=2	8	16	32	64	128	256	512	1024	budget $K\times$depth
Transformer++ (9L)	176M	8	8	8	2	—	—	—	—	—	128
GDN (9L)	161M	8	8	8	8	8†	3†	—	—	—	512
LSTM (4L)	53M	8	2	1	1	—	—	—	—	—	16
LSTM (4L)	303M	8	2	2	1	—	—	—	—	—	16–32
Looped, iter.-agnostic (2 blocks)	59M	1	1	—	—	—	—	—	—	—	—
Looped, $K\propto$ length (2 blocks)	59M	8	1	—	—	—	—	—	—	—	16
Looped, recurrent-depth (2 blocks)	82M	2	8	8†	8†	8†	8†	8†	1†	1†	2048
Full-bandwidth Tr. (1L)	8M	8	1	—	—	—	—	—	—	—	16
Full-bandwidth Tr. (9L)	180M	8	8	8†	—	—	—	—	—	—	≥128
DM (9L)	294M	8	8 (6/9 cold)	8†	8†	—	—	—	—	—	256
DM (1L)	12M	**8**	**8**	**8**†	**8**†	**8**†	**8**†	**8**†	**8**†	**8**†	**8192**

Model	params	accuracy (deepest exact depth) at n gates n=16	n=32	n=64	n=128	n=256	n=512	n=1024
Transformer++ (9L)	64M	1.00 (16)	0.90 (19)	0.70 (16)	0.62 (13)	—	—	—
Looped, $K\propto$ length	21M	1.00 (16)	0.57 (0)	0.56 (0)	—	—	—	—
Looped, iter.-agnostic	21M	1.00 (16)	1.00 (32)	1.00 (64)†	0.72 (29)†	—	—	—
Looped, recurrent-depth	30M	1.00 (16)	1.00 (32)	1.00 (64)	1.00 (128)†	0.78 (134)†	—	—
GDN	56M	1.00 (16)	0.93 (23)	0.77 (25)	0.66 (26)	—	—	—
LSTM (4L)	19M	**1.00** (16)	**1.00** (32)	0.92 (46)	0.75 (50)	—	—	—
Full-bandwidth Tr. (1L)	8M	**1.00** (16)	**1.00** (32)	**1.00** (64)	**1.00** (128)†	**1.00** (256)†	—	—
DM (1L)	12M	**1.00** (16)	**1.00** (32)	**1.00** (64)	**1.00** (128)	**1.00** (256)	**1.00** (512)	**1.00** (1024)†

Table 3: **Left:** keyed A_5 with K interleaved registers at circuit depth 8; "solved" is accuracy ≥ 0.95 at that depth, depth 1 is free retrieval. **Right:** CVP, tagged protocol, chance ≈ 0.55; each cell is in-distribution accuracy and, in parentheses, the deepest depth at which accuracy is ≥ 0.95. Every arm at its best learning rate and depth. †Warm-chained from the previous K or n. The 1024-gate DM cell used a 26k-step low-learning-rate continuation beyond the 60k budget, and the recurrent-depth loop's K=512 and 1024 cells ran at a reduced batch. Looped cells are at their best evaluation iteration count. "—" is a cell that was not run; unfinished chains are stated in the text, and the K=2 anomalies (nine-layer FBT forms in one of two seeds, recurrent-depth loop at depth 2, both forming at K=8) are unexplained. "Recurrent-depth" is the recipe of Geiping et al. (2025).

Circuit value problem. In A_5 the composition depth is the sequence length and in keyed A_5 it is our choice of D; in the circuit value problem, which is P-complete (Ladner, 1975), it is set by the instance's wiring. An instance has m=12 input bits followed by n gates in topological order. Each gate is a block $[\mathrm{id}_k,\ \mathrm{id}_a,\ \mathrm{id}_b,\ \mathrm{id}_d,\ \mathrm{id}_c]$ computing $v_k = v_c \oplus \mathrm{MAJ}(v_a, v_b, v_d)$ over three random earlier gates and the preceding gate $c = k-1$. The gate set is universal with constants, balanced, and without a single-operand shortcut (best depth-1 predictor 0.53–0.57). Since $c = k-1$ the circuit depth of gate k is k, so a token recurrence gains one level of depth per gate while a depth-L transformer must chain L lookups per position. The target at every gate is its value and values never appear in the input, so accuracy is resolved by circuit depth. Every arm gets the same size curriculum, 30k-step budget plus a 30k low-learning-rate continuation, fresh eval circuits, and its best learning rate in $\{4, 2, 1\} \times 10^{-4}$.

Table 3 (right) gives the result. A one-layer DM (11.8M) is exact, meaning accuracy ≥ 0.95 at every depth, from 16 to 1024 gates. Each lineage forms cold at 16 gates and is then chained upward, every larger cell trained at its own size from the previous solution with the vocabulary layout fixed. Three lineages reach 256, two reach 512 and one reaches 1024, where the 60k budget is exact to depth ≈ 780 and a 26k-step continuation at 2×10^{-5} completes it with minimum per-depth accuracy 0.984. The LSTM (4 layers, 18.9M) walls at depth 48–64 regardless of budget, depth or size (0.92 at 64 gates, 0.75 at 128). GDN (55.9M) walls at depth 22–28. The Transformer (9 layers, 63.7M) is exact to depth ≈ 16 at every size from 16 to 128 gates, its nine-layer reach. The iteration-agnostic loop walls at 64 gates and the instance-tied loop at 16. The recurrent-depth loop is exact at 128 gates with 64 iterations per token, with one block as well as two, and walls at 256 (exact to depth 134 with two blocks, 3 with one). The full-bandwidth transformer tracks depth like DM from one layer, exact to 256, with 512 in progress. Depth does not substitute for the recurrence: sweeping layers at d=768 moves no arm's wall, and extra DM layers do not help (App. G).

5 Comparison with looped transformers

Looping a weight-tied block is the other way to grow a transformer's depth with the input. We ran three recipes at their best learning rates: the iteration-agnostic loop with input injection and step embeddings (Bansal et al., 2022; Fan et al., 2025) at $K \sim U[K/4, K]$, the instance-tied loop with K a quarter of the sequence length (Fan et al., 2025), and the recurrent-depth loop of Geiping et al. (2025) (App. F); Fig. 3 shows the iterates.

With tuned learning rates all three recipes form A_5 at T=64 in distribution, the iteration-agnostic and recurrent-depth loops form CVP to 64 gates, the recurrent-depth loop chains to 128 gates with 64 iterations per token, with one block as well as two, and walls at 256, and with two blocks it holds keyed state to K=256 (Tables 2 and 3). In every run, formed or not, the per-iteration update shrinks (Fig. 3), and from $k \approx 32$ in every panel further iterations do not change accuracy. On A_5 evaluated at 2$\times$ the training length the deepest OOD bin sits at chance (0.02–0.06) for every iteration count from 16 to 128 in all three recipes, and the position curves reach chance by positions

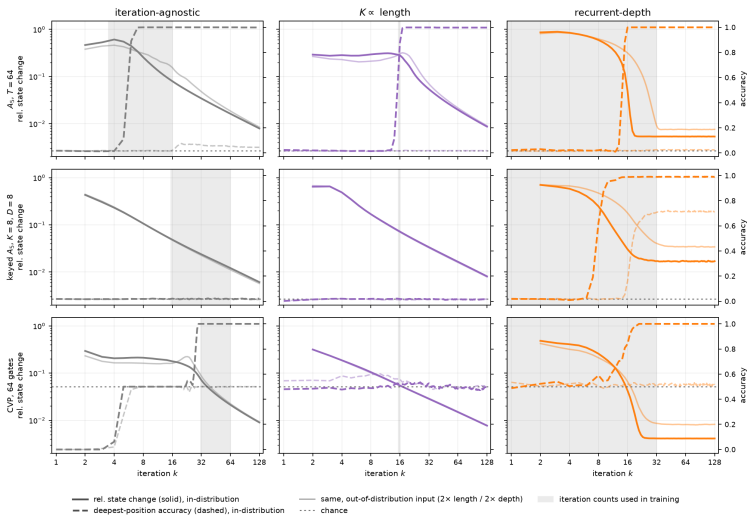


Figure 3: Looped-transformer iterates on A_5 (T=64), keyed A_5 (K=8, D=8) and CVP (64 gates). Solid is the relative change of the recurrent state per iteration (log scale), dashed is accuracy on the deepest bin at that iterate, dark for in-distribution input and light for 2× length or depth. Shading marks the iteration counts used in training, dotted is chance. The update shrinks in every panel, to a constant floor at the bf16 resolution by k=32 under the recurrent-depth recipe and as $1/k$ under the other two (the instance-tied loop only after its tied count). Formed runs settle at the answer, unformed ones (the two keyed panels on the left and the CVP instance-tied panel) at a shortcut, and once settled further iterations leave accuracy unchanged. Values in App. F.

112–160 (Fig. 2), where the LSTM stays at 1.0 and the one-layer DM holds 0.95–1.00. On CVP the 64-gate solvers score chance at the deepest gate of 128-gate circuits at every iteration count. The one exception is the recurrent-depth loop on keyed A_5, where extra iterations raise accuracy on depth-16 registers from 0.16 at k=16 to 0.71 at k=32 and no further. On the regular languages outside C-RASP all three recipes land on the Transformer's numbers at 10× the training length, and evaluating with more iterations than they were trained at changes nothing (App. E). A loop therefore grows computation only in distribution, and reaching more serial depth means retraining with a larger iteration count, at K passes per token. This matches Geiping et al. (2025) at 3.5B, where accuracy saturates by $r \approx 32$ and the latent state converges on most tokens (their §7); our measurement adds that past the trained length or depth the settled iterate is at chance.

6 Conclusion

Dual-Mod attention turns a transformer's KV tape into a non-linear recurrence. Its chunk sweeps have a unique fixed point, the sequential recurrence itself, reached in finitely many steps. Serial composition grows one level per token and each composed value is written under its own key. Serial depth comes from position, not from iterating a block. A loop's reach is fixed before training, on every tagging task we test does not extend past what it was trained on however many iterations it is given (the one partial exception is keyed A_5 in §5), and costs K passes per token, while token recurrence extends past training at one pass per token. Among token recurrences DM is the one whose exact recurrence is chunk-parallel at about a tenth of a layer per sweep, against the whole stack for full-stack feedback, and the one that reduces to the transformer when its gates close, so it composes with layer looping, with intra-layer hybrids such as NHA (Du et al., 2026), HydraHead (Tan et al., 2026) and Oryx (Li et al., 2026), and with full-stack feedback itself. It works under a sliding window, where the recurrence carries out-of-window information forward, though the growing-state results are for the full tape: under a window the addressable cache is bounded by the window even as the recurrence continues. Its length generalization is weaker than a fixed-state model's, which we trace to softmax dilution and leave fixing to future work.

ETHICS STATEMENT

This work moves serial computation out of emitted tokens and into latent state. Chain-of-thought text is a current lever for oversight, since a model that reasons in readable tokens can be monitored, and any latent-reasoning architecture, including ours, trades some of that legibility away. Two properties of DM partly offset this. The recurrent state is the KV cache itself, one addressable entry per token, so it can be read out and probed position by position rather than reconstructed from an iterated hidden trajectory, and the chunk recurrence reaches the exact sequential computation in finitely many sweeps, so there is no test-time iteration whose length or convergence is unobserved. We do not claim these make latent reasoning safe, only more inspectable than a fixed carry or a loop. The models in this paper are at most 1.3B parameters, trained on public corpora and synthetic formal-language tasks, and we do not foresee misuse beyond that of language models of this size.

REPRODUCIBILITY STATEMENT

Code for the architecture, the chunk-parallel training engine, the state-tracking task generators, the looped and full-bandwidth baselines, and the evaluation harnesses is available at `https://anonymous.4open.science/r/dual_mod_public-41F1/`, together with the training configurations, seeds and logs behind every table and the checkpoints of the formed solvers (the parity, A_5, keyed A_5 and CVP lineages). Task definitions, the tuning protocol for baselines and all hyperparameters are given in the appendix.

AI USE STATEMENT

Large language models were used in this work. They have helped in designing and running experiments, writing and running the training and evaluation code, the task generators for the synthetic state-tracking datasets, and the analysis scripts. They have been used for literature review for finding and checking related work. They have helped draft some sections of the paper and polished the writing. They have assisted with the mathematical claims, including the finite-step convergence and circuit-class arguments in the method and related-work sections.

REFERENCES

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *ArXiv*, abs/2402.12875, 2024. URL https://api.semanticscholar.org/CorpusID:267760184.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 2022. URL https://api.semanticscholar.org/CorpusID:257050206.

Sophie Hao, Dana Angluin, and Roberta Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10: 800–810, 2022. URL https://api.semanticscholar.org/CorpusID:248177889.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2019. URL https://api.semanticscholar.org/CorpusID:189928186.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=din0lGfZFd.

Qwen Team. Qwen Studio — qwen.ai. https://qwen.ai/blog?id=qwen3.8, 2026. [Accessed 06-09-2026].

Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, and Y. Charles. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models, 2019. URL https://arxiv.org/abs/1909.01377.

Yifan Zhang, Wenyu Du, Dongming Jin, Jie Fu, and Zhi Jin. Finite state automata inside transformers with chain-of-thought: A mechanistic study on state tracking. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13603–13621, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.668. URL https://aclanthology.org/2025.acl-long.668/.

Nanbeige Lab, Chen Yang, Chengrui Huang, Fufeng Lan, Hanhui Chen, Hao Zhou, Huatong Song, Jiaqi Cao, Jiaying Zhu, Jinlin Niu, Kai Wang, Lisheng Huang, Qiliang Liang, Ran Le, Ruixiang Feng, Shuang Sun, Tao Gu, Tao Zhang, Tianyu Luo, Yang Song, Yun Xing, Yuntao Wen, Ziyao Xu, Zongchao Chen, and Zongqiang Li. Nanbeige4.2-3b: Unlocking agentic capabilities in a compact model, 2026. URL https://arxiv.org/abs/2607.22083.

Richard E. Ladner. The circuit value problem is log space complete for p. *SIGACT News*, 7(1):18–20, January 1975. ISSN 0163-5700. doi: 10.1145/990518.990519. URL https://doi.org/10.1145/990518.990519.

David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. In *Symposium on the Theory of Computing*, 1986. URL https://api.semanticscholar.org/CorpusID:9031549.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=QZgo9JZpLq.

Riccardo Grazzi, Julien Siems, Jörg K.H. Franke, Arber Zela, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear RNNs through negative eigenvalues. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=UvTo3tVBk2.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=PPjSKy40XUB.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2edigk8yoU.

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J Zico Kolter, and Roger Baker Grosse. Path independent equilibrium models can better exploit test-time computation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=kgT6D7Z4Xv9.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

Chengting Yu, Xiaobo Shu, Yadao Wang, Yizhen Zhang, Haoyi Wu, Jiaang Li, Rujiao Long, Ziheng Chen, Yuchi Xu, Wenbo Su, and Bo Zheng. MeSH: Memory-as-state-highways for recursive transformers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IhTrFvY7p3.

Wanyun Cui. A hippocampus for linear attention: An exact memory for what the recurrent state forgets, 2026. URL https://arxiv.org/abs/2607.02303.

Leon Lufkin, Tomás Figliolia, Beren Millidge, and Kamesh Krishnamurthy. Hybrid associative memories, 2026. URL https://arxiv.org/abs/2603.22325.

Zhentao Tan, Wei Chen, Jingyi Shen, Yao Liu, Xu Shen, Yue Wu, and Jieping Ye. Hydrahead: From head-level functional heterogeneity to specialized attention hybridization, 2026. URL https://arxiv.org/abs/2606.20097.

Kevin Y. Li, Asher Trockman, Ananda Theertha Suresh, and Ziteng Sun. Multi-mixer models: Flexible sequence modeling with shared representations, 2026. URL https://arxiv.org/abs/2605.28769.

Jusen Du, Jiaxi Hu, Tao Zhang, Weigao Sun, and Yu Cheng. Native hybrid attention for efficient sequence modeling, 2026. URL https://arxiv.org/abs/2510.07019.

Songlin Yang, Yikang Shen, Kaiyue Wen, Shawn Tan, Mayank Mishra, Liliang Ren, Rameswar Panda, and Yoon Kim. PaTH attention: Position encoding via accumulating householder transformations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ZBlHEeSvKd.

Mingyu Xu, Tenglong Ao, Jiaao He, Jianqiao Lu, Guang Shi, and Shu Zhong. Deltaformer: Unlock the state space of transformer. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=GSE3oaiDL2.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Increasing the expressivity of deltanet through products of householders. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025. URL https://openreview.net/forum?id=nvb60szj5C.

Xi Wang, Ziyang Cai, Zheng Zhan, Harry Dong, Ying Fan, Gustavo de Rosa, Tim Pearce, and John Langford. Full-bandwidth transformer, 2026. URL https://arxiv.org/abs/2608.08888.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. Addressing some limitations of transformers with feedback memory, 2021. URL https://arxiv.org/abs/2002.09402.

Ziyang Cai, Xingyu Zhu, Yihe Dong, Yinghui He, and Sanjeev Arora. T^2MLR: Transformer with temporal middle-layer recurrence, 2026. URL https://arxiv.org/abs/2607.15178.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:13756489.

Neehal Tumma, Noel Loo, and Daniela Rus. Preconditioned deltanet: Curvature-aware sequence modeling for linear recurrences. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=UC6YiTOeKb.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=e93ffDcpH3.

Simran Arora, Aman Timalsina, Aaryan Singhal, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Re. Just read twice: closing the recall gap for recurrent language models. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, 2024b. URL https://openreview.net/forum?id=RsqCqziLAt.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated deltanet-2: Decoupling erase and write in linear attention, 2026. URL https://arxiv.org/abs/2605.22791.

Andy Yang, Blerta Veseli, Corentin Barloy, Michaël Cadilhac, Andreas Krebs, Charles Paperman, Howard Straubing, and Michael Hahn. Algebraic decomposition theory for transformer length generalization, 2026. URL https://arxiv.org/abs/2608.13433.

David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention, 2022. URL https://arxiv.org/abs/2202.12172.

A SUMMARY OF PROPERTIES

	Transf.	Linear rec.	LSTM	Looped	FBT	**DM**
Exact superset of self-attention	✓	✗	✗	✓[a]	✗[b]	✓
Serial depth grows with the input at one pass per token	✗	✗	✓	✗[c]	✓	✓
State grows with the context at fixed width	○	✗	✗	○[d]	✓[e]	✓
Reach not capped by a trained iteration count	✓	✓	✓	✗	✓	✓
Length-generalizes outside C-RASP	✗	✗	✓	✗	✓	○[f]
Exact recurrence trains in parallel	✓	✓	✗	✓[g]	○[h]	✓
Per-token inference cost of a transformer	✓	✓	✓	✗[g]	✓	✓
Competitive language modeling at ≥1B	✓	✓	✗	✓	✓[h]	✓

Table 4: What each family provides. Columns describe the recurrent component; hybrids inherit the transformer's rows through their attention layers, not through their recurrence. [a]At $K{=}1$. [b]The token enters only through the gate on the carried state. [c]Depth is K passes per token, set before training. [d]The recurrent-depth loop holds keyed state to $K{=}256$ with two blocks at seven times DM's size and none with one block, and walls on the depth axis at 256 gates. [e]Holds depth 8 at $K{=}2$, 8 and 16 with nine layers and 180M parameters, and only $K{=}2$ with one layer. [f]Holds to 8–10$\times$ the training length, then decays (App. E). [g]K full passes per token in training and inference, and a cache per iteration. [h]The exact recurrence trains sequentially in T, or at about $25\times$ the parallel FLOPs under the chunk scheme of §3.2 against DM's $3.3\times$; the few-pass approximation trained at scale forms none of our tasks at $\leq 4\times$ and gains reach only with the pass count (App. F).

B TASK DEFINITIONS

Tagging protocol. Every state-tracking task follows Merrill et al. (2024). The input stream contains only operations, the target at every position is the running state after that operation, and instances are freshly sampled from the generator rather than drawn from a fixed set, with evaluation sampled independently. States never appear in the input. An earlier interleaved format that placed each prefix state in the stream beside its operation was abandoned because it reduces state tracking to a depth-1 lookup of the previous state, and a fixed-depth transformer solves it at $4\times$ the training length. A model must therefore produce the state at position t from the operations alone, with no intermediate state to copy.

A_5 word problem. Operations are elements of A_5 drawn uniformly, the state is the running product, and chance is $1/60$. Training is at $T{=}64$ with 12k steps unless stated, and evaluation at 64, 128 and 256 positions. "Deepest" is accuracy on positions 57–64. The short grid for the minimum-depth sweep uses $T \in \{5, 10, 15, 20\}$.

Keyed A_5. K independent A_5 running products are interleaved in one stream, D operations per register in a random interleaving. Each operation token carries a register tag and a group element, and its target is that register's updated running product, the composition of every operation the register has received so far; the depth of a position is that count, from 1 to D. After the operations each register is queried once and the target is its final product, at depth D. A solution must hold the current state of every register and compose the queried one to its depth. Depth 1 is retrieval of the register's first element and is excluded from scoring. A register depth counts as solved at accuracy ≥ 0.95, contiguous from depth 2, and an architecture's budget is K times its deepest solved depth. All main-text results use $D{=}8$ and a 12k-step budget. Cells beyond an architecture's cold reach are warm-started from its previous solver and grown by a register ramp, with key-slot shuffling per batch so that every register slot trains and the new key embeddings initialized from the mean of the trained ones. GDN and DM are given this recipe identically.

Circuit value problem. An instance has $m{=}12$ input bits followed by n gates in topological order. Each gate is a block $[\mathrm{id}_k,\ \mathrm{id}_a,\ \mathrm{id}_b,\ \mathrm{id}_d,\ \mathrm{id}_c]$ computing $v_k = v_c \oplus \mathrm{MAJ}(v_a, v_b, v_d)$ over three random earlier gates and the preceding gate $c = k - 1$, so the circuit depth of gate k is k; the gate's value is committed at the last of its five tokens. The target at every gate is its value and values

never appear in the input. The gate set with constants is universal and keeps values balanced. Before use, the format was certified against two shortcuts. The marginal of every gate value is within a point of 0.5, and the best depth-1 predictor from any single operand scores 0.53–0.57, which is the chance line in Table 3. Every arm gets the same size curriculum, gates and inputs ramping from 4 over the first half of the run, a 30k-step budget plus a 30k low-learning-rate continuation, and fresh evaluation circuits. A cell is exact when per-depth accuracy is ≥ 0.95 at every depth. Larger sizes are chained, each cell trained at its own size from the previous solution with the vocabulary layout fixed so that no token id moves.

Length generalization. The regular-language experiments use the DFA state-prediction protocol of Yang et al. (2026), described in App. E.

C Language Modeling

C.1 Language Modeling Results at 340M

Tables 6–8 give the full 340M results summarized in §4.1. All rows marked *Ours* are trained on one shared token stream under the recipe of Tumma et al. (2026), so the Transformer++ and GDN rows are the parameter-matched controls for every claim about DM's own margins; Du et al. (2026) use the same data and token count with a different learning-rate schedule and 30,720 steps. Table 5 gives the test and val loss compared with baselines.

Model	d	non-embed. params	val CE ↓	test CE ↓
Transformer++	1024	303.6M	2.4816	2.5358
GDN	1024	334.0M	2.4672	2.5217
DM	960	329.6M	**2.4399**	**2.4946**

Table 5: Cross-entropy at 340M on the shared SlimPajama stream after 15B tokens, validation and held-out test splits. DM's width is reduced to 960 so that its non-embedding count, including the DM-specific projections, stays inside the Transformer++–GDN band.

	LM ppl ↓		**Commonsense Reasoning ↑**									
Model	Wiki	LMB	LMBa	ARC_e	ARC_c	HellaS	PIQA	WinoG	Avg_6	BoolQ	SciQ	Avg_8
Tumma et al. (2026)												
DeltaNet	29.04	45.76	29.58	44.02	23.55	34.08	65.07	50.91	41.20	56.80	75.30	47.41
PDN	29.05	43.05	30.48	44.70	23.52	33.83	65.14	52.41	41.68	58.50	75.10	47.96
GDN	27.08	31.39	34.50	44.49	24.32	35.96	65.83	51.30	42.73	56.30	78.20	48.86
PGDN	26.86	28.43	35.55	47.81	24.21	36.19	64.83	**52.97**	43.59	60.10	78.00	49.96
KDA	26.18	31.37	34.04	45.45	22.70	36.06	66.00	52.25	42.75	57.00	**79.50**	49.12
PKDA	25.98	**25.33**	**37.22**	46.97	22.95	**37.01**	65.68	51.46	43.55	59.10	78.70	49.89
Cui (2026)												
HOLA+recency	25.04	32.33	34.72	46.13	24.40	35.62	65.34	52.80	43.17	—	—	—
HOLA	**22.92**	30.26	34.54	46.00	24.06	35.91	65.02	51.54	42.85	—	—	—
Du et al. (2026)												
Mamba2-H	25.81	43.35	30.86	**48.23**	**24.49**	36.31	65.23	48.62	42.29	—	—	—
GLA-H	27.94	40.12	31.83	45.20	24.32	34.42	63.93	49.96	41.61	—	—	—
GSA-H	26.19	46.05	30.91	45.54	23.72	35.19	65.23	50.99	41.93	—	—	—
GDN-H	25.67	51.79	29.30	45.75	23.21	35.80	65.56	51.38	41.83	—	—	—
NHA	25.97	38.38	32.72	47.69	24.32	36.03	65.67	52.09	43.09	—	—	—
Ours												
Transformer++	26.99	36.21	33.44	45.58	23.55	35.43	65.18	51.07	42.37	60.61	78.50	49.17
GDN	26.57	29.51	34.68	45.71	23.63	36.63	65.51	51.38	42.92	59.27	78.30	49.39
DM	25.42	30.68	35.80	46.38	**24.49**	36.88	**66.65**	51.70	**43.65**	**60.70**	78.40	**50.12**

Table 6: Language modeling and commonsense evaluation in the 340M-param / 15B-token setting. Our rows are trained on a single shared token stream. Avg_6 is the mean of the six columns preceding it, following Cui (2026) and Du et al. (2026); Avg_8 adds BoolQ and SciQ, following Tumma et al. (2026).

Model	FDA	SWDE	SQuAD	Avg_3
Tumma et al. (2026)				
DeltaNet	8.53	27.09	28.32	21.31
PDN	14.16	29.17	28.89	24.07
GDN	13.61	29.43	29.83	24.29
PGDN	14.25	32.24	31.76	26.08
KDA	13.88	34.11	29.69	25.89
PKDA	14.25	34.65	31.39	26.76
Ours				
Transformer++	17.60	48.96	13.74	26.77
GDN	16.42	27.72	31.03	25.06
DM	**26.50**	**52.57**	**41.86**	**40.31**

(a) Based protocol (Arora et al., 2024a)

Model	FDA	SWDE	SQuAD	TQA	NQ	DROP	Avg_6
Du et al. (2026)							
Transformer++	46.14	25.87	33.22	45.97	18.94	20.03	31.70
Mamba2-H	55.68	39.55	27.98	45.50	18.34	17.63	34.11
GLA-H	58.86	39.36	31.37	44.14	17.01	18.21	34.83
GSA-H	**62.13**	45.36	31.17	43.78	**20.62**	18.78	36.97
GDN-H	52.32	38.80	29.83	39.40	15.24	17.92	32.25
NHA	53.86	**48.55**	**38.60**	46.56	20.56	**23.48**	**38.60**
Ours							
Transformer++	48.91	35.82	28.72	44.08	18.75	21.03	32.89
GDN	22.96	26.19	28.25	43.78	14.60	19.45	25.87
DM	52.99	43.70	34.18	**47.51**	20.40	20.03	36.47

(b) JRT protocol (Arora et al., 2024b)

Table 7: Zero-shot in-context retrieval at 340M, reported under the protocol each source used (left: Based; right: JRT). Avg_3 averages FDA, SWDE and SQuAD, following Tumma et al. (2026); Avg_6 averages all six tasks, following Du et al. (2026). The NHA-paper Transformer++ row is a calibration reference for ours. Zero-shot recall at this scale is strongly harness-dependent so we compare only within a protocol.

	MQAR key–value pairs ↑				
Model	8	16	32	64	Avg
Transformer++	37.11	29.69	13.28	9.38	22.46
GDN	50.39	44.53	28.71	14.84	34.64
DM	**65.23**	**66.41**	**57.62**	**43.75**	**58.25**

(a) Zero-shot associative recall

	S-NIAH-1			S-NIAH-2			S-NIAH-3			MK-NIAH-1				MK-NIAH-2			MK-NIAH-3		
Model	2K	4K	8K	2K	4K	8K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	1K	2K	4K
Transformer++	99.4	0.0	0.0	**100.0**	0.0	0.0	64.4	0.0	0.0	69.4	53.4	0.0	0.0	17.2	7.8	0.0	24.6	**7.4**	**2.4**
GDN	**100.0**	**100.0**	**98.8**	99.0	**43.2**	**16.0**	71.6	**42.4**	**15.6**	21.4	19.8	**21.6**	**8.2**	1.2	0.2	0.0	0.0	0.0	0.0
DM	**100.0**	23.6	2.0	**100.0**	36.4	2.0	**84.6**	2.2	0.0	**83.8**	**75.2**	12.0	0.2	**29.6**	**14.8**	0.0	**31.2**	3.2	0.0

(b) RULER needle-in-a-haystack

Table 8: Recall at 340M on our shared token stream. (a) MQAR, 64 sequences per load, zero-shot from LM pretrain; 1500 steps of MQAR finetuning drives every architecture to 1.000 at all loads. (b) RULER NIAH, 500 samples per cell, exact-length prompts; all models are trained at 2K.

C.2 Published 1.3B hybrids on other corpora

Table 9 places the 1.3B hybrids of Du et al. (2026), trained on SlimPajama with a different tokenizer, beside the FineWeb-Edu rows of Table 1. The two blocks are not comparable cell for cell, so marks are assigned within each block separately. On retrieval NHA scores 46.43 against 45.64 for the windowed DM, with its margin concentrated on FDA and SWDE, the two semi-structured extraction tasks, while the FineWeb-Edu rows match or lead on the four reading-comprehension tasks.

	LM ppl ↓		Commonsense Reasoning ↑							In-context Retrieval (JRT, 2K) ↑						
Model	Wiki	LMB	LMBa	PIQA	HellaS	WinoG	ARC_e	ARC_c	Avg_6	FDA	SWDE	SQuAD	TQA	NQ	DROP	Avg_6
Du et al. (2026), SlimPajama																
Transformer++	17.61	15.86	45.60	70.08	49.21	56.27	55.01	28.07	50.71	44.32	32.43	42.59	58.47	24.49	21.56	37.31
Mamba2-H	**16.00**	12.77	48.46	71.93	**52.73**	55.49	57.74	28.24	52.43	65.30	50.89	41.99	55.92	28.54	22.46	44.18
GLA-H	17.58	15.72	46.71	70.40	48.71	54.46	56.73	27.56	50.76	**69.48**	46.77	40.28	57.05	27.15	21.47	43.70
GSA-H	16.22	23.09	41.92	71.71	51.70	53.35	57.79	28.07	50.76	68.30	51.83	42.36	58.59	26.42	22.42	44.99
GDN-H	16.02	**11.68**	**49.78**	70.57	52.52	56.27	58.12	27.99	52.54	66.76	49.02	41.15	59.06	**29.30**	24.01	44.88
NHA	16.16	12.58	49.19	**72.03**	52.08	**56.83**	**58.71**	**28.50**	**52.89**	68.30	**52.48**	**45.58**	**59.60**	27.18	**25.44**	**46.43**
Ours, FineWeb-Edu																
DM (+2K SWA)	16.14	10.61	50.32	73.07	59.59	58.17	72.69	36.52	58.39	63.25	49.41	45.84	65.23	27.21	22.90	45.64
DM (full 4K attention)†	15.96	9.94	51.99	72.85	59.26	58.56	72.10	38.14	58.82	70.24	58.24	49.16	66.47	30.16	23.77	49.67

Table 9: 1.3B parameters, 100B tokens. The top block is trained on SlimPajama with its own tokenizer, the bottom block on FineWeb-Edu with the Llama-2 tokenizer under the GDN-2 recipe, so the blocks are not comparable cell for cell; marks are assigned within the top block only. Retrieval uses the JRT harness at 2K (Arora et al., 2024b). †Full 4K attention with no window, shown for reference.

C.3 RULER at 1.3B

Table 10 gives the needle-in-a-haystack results summarized in §4.1. The quoted rows are the GDN-2 paper's numbers under its own protocol, and ours use the official RULER generator with the Llama-2 tokenizer and 500 samples per cell, so the two blocks are close but not identical in prompt construction. The windowed DM extrapolates past its 2K window on the single-needle tasks at the

hybrids' level, and at 8K retains 22 to 24 where the windowed Transformer under the same recipe scores zero. The full-attention variant leads every 4K column and every MK-NIAH-1 column, trails the hybrids on S-NIAH-3 at 1K and 2K, and drops to zero beyond its training length, like a plain transformer. Table 11 adds MK-NIAH-2, MK-NIAH-3, MQ and MV, for which no baseline at this scale has been published.

	S-NIAH-1				**S-NIAH-2**				**S-NIAH-3**			**MK-NIAH-1**		
Model	1K	2K	4K	8K	1K	2K	4K	8K	1K	2K	4K	1K	2K	4K
Hatamizadeh et al. (2026)														
Transformer	**100.0**	**100.0**	51.2	0.0	**100.0**	**100.0**	44.2	0.0	95.8	94.8	37.0	75.6	66.6	38.2
Mamba-2	**100.0**	**100.0**	51.8	25.4	**100.0**	99.6	52.4	25.8	97.8	86.8	48.0	82.0	58.6	30.6
Gated DeltaNet	**100.0**	**100.0**	47.2	22.4	**100.0**	99.8	57.3	25.6	94.8	91.2	47.2	91.0	78.4	44.8
KDA	**100.0**	**100.0**	51.8	26.2	**100.0**	**100.0**	56.0	23.0	97.2	93.4	51.6	91.4	84.0	40.4
Mamba-3 (SISO)	**100.0**	**100.0**	49.6	26.0	**100.0**	**100.0**	**58.2**	27.8	95.0	90.4	44.0	78.8	65.6	33.6
Mamba-3 (MIMO)	**100.0**	**100.0**	47.8	22.0	**100.0**	**100.0**	53.0	27.8	99.4	98.4	54.2	82.4	79.0	40.6
Gated DeltaNet-2	**100.0**	**100.0**	**55.2**	**27.4**	**100.0**	**100.0**	57.9	**29.2**	**99.6**	**99.0**	**55.6**	**93.0**	84.6	**48.0**
Ours														
DM (+2K SWA)	**100.0**	89.2	53.2	23.8	**100.0**	**100.0**	51.0	22.0	87.0	70.4	42.8	91.8	**87.6**	42.2
DM (full 4K attention)†	100.0	99.8	100.0	0.0	100.0	99.8	82.8	0.2	87.2	83.2	77.2	98.6	98.2	70.6

Table 10: RULER needle-in-a-haystack at 1.3B; all models are trained at 4K with a 2K sliding window. Quoted rows are under the GDN-2 paper's protocol; ours use the official RULER generator (Llama-2 tokenizer, 500 samples per cell, `string_match_all`). †Full quadratic attention over 4K with no window; shown for reference and excluded from marking.

	MK-NIAH-2			**MK-NIAH-3**			**MQ-NIAH**			**MV-NIAH**		
Model	1K	2K	4K	1K	2K	4K	1K	2K	4K	1K	2K	4K
DM (+2K SWA)	69.8	29.2	18.0	42.2	14.6	10.4	92.5	79.2	27.0	89.5	72.0	22.0
DM (full 4K attention)†	94.0	86.2	59.4	52.8	39.6	27.6	93.1	77.8	29.8	96.4	67.6	34.0

Table 11: RULER multikey-2, multikey-3, multiquery and multivalue needle-in-a-haystack at 1.3B, official RULER generator, Llama-2 tokenizer, 500 samples per cell, `string_match_all`. No baseline at this scale has been published for these four tasks. †Full quadratic attention over 4K with no window.

C.4 Training operator versus exact recurrence at 1.3B

The 1.3B model is trained and scored at $K{=}24$ sweeps on 64-token chunks, below the $K{=}c$ at which the chunk recurrence is exact. Table 12 rescores the same full-attention checkpoint (step 190,649) at $K{=}c{=}64$. Cross-entropy moves by 0.001, the nine-task commonsense average by 0.01 with every task within 0.6, four of six retrieval tasks by under a point, TQA by 1.2, and FDA by 3.8 in the exact direction. During training the $K{=}64$ validation loss tracked the operating-K loss to within 10^{-3} at every evaluation.

In the engine, a committed chunk is frozen in the sense that the forward pass never refines its entries again; it is not detached. Every later read of a committed key or value returns its gradient into the cache, and the writer's backward hands those gradients to the sweeps that produced the entry, so the training gradient is the exact adjoint of the forward computation with no truncation. At $K{=}c$ the engine reproduces the exact sequential scan: in fp32 the loss agrees to 10^{-6}, the logits to 3×10^{-5} and every parameter-gradient group to a relative error below 10^{-5} ($T{=}8192$, $c{=}64$, two layers, full causal attention); in bf16 the residual is accumulation noise (loss 2×10^{-4}, logits 8×10^{-2} max).

	val CE ↓	Wiki ↓	LMB ↓	Avg_9	SWDE	SQuAD	FDA	TQA	NQ	DROP	Avg_6
K=24 (train)	2.0285	15.96	9.94	54.25	58.2	49.2	70.2	66.5	30.2	23.8	49.7
K=64 (exact)	2.0294	16.00	10.07	54.24	58.4	48.9	74.0	65.3	30.1	24.0	50.1

Table 12: The 1.3B full-attention DM checkpoint scored with its training operator (K=24) and with the exact chunk recurrence (K=c=64). Held-out FineWeb-Edu cross-entropy, LM perplexities, the nine-task commonsense average of Table 1 and the six JRT retrieval tasks at 2K.

D Ablations

D.1 The k-mod head and respelling

A drawback of DM shows up in verbatim retrieval of needles. The model learns to treat unimportant tokens as registers and weights the retrieved $\boldsymbol{u}_t$ above the raw token when writing them ($\boldsymbol{g} \ll 0.5$). But in many NIAH tasks the importance of a token is only known in the future. A UUID starts as what looks like garbage in a causal stream, and only later does it become clear that those tokens are a needle that will have to be reproduced. Since the recurrence is causal there is no way to go back and restore the blended values to raw values. The effect is a hysteresis in the value tape that makes the early part of a UUID hard to respell verbatim. Finding the needle appears not to be the problem, since a k-mod head still locates it, though we do not measure localization separately. The fix we use is to leave a single head as k-mod rather than dual-mod, so that one head always preserves the raw value tokens and the other heads can fetch raw content through it. Table 13 shows the trade. The k-mod head is neutral on language and cross-entropy, costs associative recall on MQAR, and buys verbatim recall, with the largest gains on the UUID needle and the multikey tasks. The one needle it does not help is multikey 3, which collapses past 1k for every architecture.

	k-mod head	**no k-mod**	Δ
Commonsense and loss			
Avg_6	43.65	**43.95**	+0.30
Avg_8	50.12	**50.14**	+0.02
val CE ↓	2.440	**2.438**	−0.002
Natural retrieval (JRT)			
Avg_6	**36.47**	36.19	−0.28
FDA / SWDE	53.0 / **43.7**	**56.2** / 41.6	+3.2 / −2.1
SQuAD / NQ	34.2 / **20.4**	**35.2** / 18.6	+1.0 / −1.8
TQA / DROP	**47.5** / **20.0**	46.4 / 19.3	−1.1 / −0.8
MQAR (pairs)			
8 / 16 / 32 / 64	65.2 / 66.4 / 57.6 / 43.8	**72.3** / **69.9** / **59.0** / **51.0**	+7.1 / +3.5 / +1.4 / +7.2
RULER NIAH at 2K			
S-NIAH-1 / 2	100 / **100**	100 / 97.4	0 / −2.6
S-NIAH-3 (UUID)	**84.6**	38.6	−46.0
MK-NIAH-1	**75.2**	55.6	−19.6
MK-NIAH-2	**14.8**	10.6	−4.2
mean (5 tasks)	**74.9**	60.4	−14.5
RULER NIAH at 1K			
MK-NIAH-1 / 2	**83.8** / **29.6**	60.4 / 22.0	−23.4 / −7.6
MK-NIAH-3 (UUID→UUID)	31.2	**47.8**	+16.6

Table 13: k-mod head ablation at 340M, d=960, 26 layers, 30k steps, identical seed and data. Accuracy in %, Δ is no k-mod minus k-mod head. Natural retrieval is the six-task JRT protocol (Arora et al., 2024b), NIAH is RULER with 500 samples per cell at exact prompt length. All arms collapse past 2K, and MK-NIAH-3 collapses past 1K for every architecture.

D.2 LOW-RANK WEIGHTS

The DM-specific weights do not need to be full $d \times d$ matrices. We first trained a full rank model and inspected its contextual projections to see how compressible each learned matrix is. Table 14 gives the singular-value analysis of the four 896^2 operators in the full-rank d896 mult-res model at step 24750, over all 26 layers, using the Gavish–Donoho hard threshold to separate signal from the Adam-diffusion floor. Every operator's signal rank sits under $d/8{=}112$ on average. A few layers peak near 117 and some $\boldsymbol{W}_v$ layers are near rank 1.

Operator	signal rank, mean [min..max]	Frobenius energy beyond rank 112
$\boldsymbol{W}_k$ (key ctx)	65 [44..117]	56%
$\boldsymbol{W}_v$ (value ctx)	53 [1..91]	51%
$\boldsymbol{G}$ (mult-res gate)	74 [38..107]	62%
$\boldsymbol{H}$ (mult-res map)	86 [42..116]	62%

Table 14: Signal rank of the full-rank contextual operators in the d896 mult-res model, 26 layers, estimated under the Gavish–Donoho threshold; $d/8{=}112$. The last column is the fraction of squared singular-value mass in components beyond rank 112, which the threshold attributes to the Adam-diffusion floor.

Factoring all four operators at rank $d/8$ frees 62.6M parameters at d896, while widening the trunk from 896 to 960 costs 55.3M, so the widening fits inside the full-rank budget with room to spare. Table 15 depicts this. The full-rank d832 model, the d896 model with low-rank gates only, and the d960 model with every DM operator at rank $d/8$, which is the flagship. With 19M fewer parameters than the full-rank d896, the d960 model is better on loss, both perplexities and the six-task composite, and every NIAH needle at 2K, and trails it only on ARC-e and FDA by under a point. At this scale the contextual operators of a trained DM carry at most about $d/8$ of signal, so the budget is better spent on width than on rank; the three arms also differ in depth and operator, so this is an allocation result rather than an isolated rank sweep. The 1.3B models use rank $d/16$ on the same reasoning.

Arm	$d \times L$	ϕ	ctx rank	gate rank	non-embed	val CE ↓	test CE ↓	Wiki ↓	LMB ↓	Avg_6
d832, full rank	832×24	linear	full	full	300.4M	2.4792	2.5346	27.01	32.39	43.29
d896, low-rank gates	896×26	mult-res	full	112	348.3M	2.4595	2.5154	26.25	37.12	42.73
d960, rank $d/8$ (flagship)	960×26	mult-res	120	120	329.6M	**2.4399**	**2.4946**	**25.42**	**30.68**	**43.65**

Table 15: Rank versus width at 340M, every arm on the shared token stream. The d832 row is the K-norm arm, the best of the four full-rank d832 recipes on loss. The d896 model is scored at its best-validation checkpoint (step 26250).

D.3 NONLINEARITIES

The modded pair in Eq. 4 is built from two write operators that form the *mult-res* protocol. In general form the equations can be considered as

$$\boldsymbol{k}_t^{\text{ctx}} = \theta(\boldsymbol{u}_t), \qquad \boldsymbol{v}_t^{\text{ctx}} = \phi(\boldsymbol{u}_t, \boldsymbol{x}_t), \tag{5}$$

where $\boldsymbol{u}_t$ is the normalized attention read and $\boldsymbol{x}_t$ the layer input, with θ producing the contextual key and ϕ the contextual value that the gates blend with the raw pair. The *mult-res* protocol uses a normalized linear θ in the form of $\lambda\,\mathrm{RMSNorm}(\boldsymbol{W}_k\boldsymbol{u}_t)$, and the mult-res ϕ as $\boldsymbol{W}_v(\boldsymbol{u}_t + \sigma(\boldsymbol{G}\boldsymbol{x}_t) \odot \tanh(\boldsymbol{H}\boldsymbol{u}_t))$. Table 16 is the sweep over θ and ϕ behind that choice, run at 63M ($d{=}768$, 9 layers) on the tagging protocol at $T{=}64$ and on 10k-step runs on the ClimbMix dataset for language modeling loss and recall. Every value-side candidate is a map from $\boldsymbol{u}_t$ and the normalized input $\hat{\boldsymbol{x}}_t$ to a pre-projection $\boldsymbol{v}_{\text{in}}$, which $\boldsymbol{W}_v$ then projects. They differ in whether the map is bounded and whether it keeps an identity path from $\boldsymbol{u}_t$. The key side is a normalized linear projection in every row but one, which makes it an MLP and keeps the value linear, so that row is not an isolated test of key-side nonlinearity against mult-res. The sweep is matched-seed but single-seed. Formation on state-tracking tasks is a seed lottery for every operator, so a cell that did not form says that the operator did not form at that budget and seed, not that it cannot.

Operator	key path	value pre-projection $\boldsymbol{v}_{\text{in}}$	identity path	$\mathbb{Z}_{10}$, 12k in-dist	$\mathbb{Z}_{10}$, 12k 2×	A_5 staged in-dist	A_5 staged 2×	A_5 staged 4×	ClimbMix dev val CE ↓	ClimbMix dev MQAR@8
linear	linear	$\boldsymbol{u}$	—	0.88[a]	0.50	0.05	—	—	3.416	0.03
mult	linear	$\sigma(\boldsymbol{G}\hat{\boldsymbol{x}}) \odot \tanh(\boldsymbol{H}\boldsymbol{u})$	no	**1.00**	**0.82**	**1.00**	**0.83**	**0.43**	3.412	0.15
mult-res	linear	$\boldsymbol{u} + \sigma(\boldsymbol{G}\hat{\boldsymbol{x}}) \odot \tanh(\boldsymbol{H}\boldsymbol{u})$	yes	0.47–0.70	0.29–0.40	**1.00**	0.73	0.38	**3.409**	**0.31**
fusion	linear	$\boldsymbol{u} + \text{silu}(\boldsymbol{W}[\hat{\boldsymbol{x}}; \boldsymbol{u}])$	yes	0.87	0.62	**1.00**	0.71	0.37	**3.409**	0.09
pure MLP	linear	$\text{silu}(\boldsymbol{W}[\hat{\boldsymbol{x}}; \boldsymbol{u}])$	no	**1.00**	0.81	**1.00**	0.61	0.32	—	—
GLU	linear	$\text{silu}(\boldsymbol{W}_g[\hat{\boldsymbol{x}}; \boldsymbol{u}]) \odot \boldsymbol{W}_m[\hat{\boldsymbol{x}}; \boldsymbol{u}]$	no	0.54	0.48	—	—	—	—	—
value MLP	linear	$\boldsymbol{u} + \text{silu}(\boldsymbol{W}\boldsymbol{u})$	yes	0.54	0.43	—	—	—	—	—
key MLP	MLP	$\boldsymbol{u}$	—	0.13	—	—	—	—	—	—

Table 16: Write-operator sweep at 63M, tagging protocol at T=64, one seed per cell with the seed matched across rows. $\hat{\boldsymbol{x}}$ is the normalized layer input and every $\boldsymbol{v}_{\text{in}}$ is followed by $\boldsymbol{W}_v$. Chance is 0.10 on $\mathbb{Z}_{10}$ and 0.017 on A_5. "Staged" grows the operator from a formed linear skeleton. ClimbMix runs are 10k steps, MQAR zero-shot at 8 pairs. [a] 0.999 at 40k steps with a spike guard.

The two things that we see from this ablation are the following. (1) The key path should carry no extra learned non-linearity. The key-MLP variant did not track a mod-10 sum under this configuration and budget, and the likely reason is that the value non-linearity already reaches the key one write-read hop later through $\boldsymbol{u}$, so a second one corrupts the anchor that later reads match against. (2) In this sweep the value must be non-linear. The linear value is the slowest to form on $\mathbb{Z}_{10}$, the only operator that fails to form on A_5, and the worst on both loss and recall on the dev language modeling task. Among bounded value non-linearities, state tracking does not discriminate. Staged from a formed skeleton, mult, mult-res, fusion and pure MLP all reach 1.00 at every position, and plain mult extrapolates best. The choice of mult-res comes from the language modeling task. At tied loss it doubles plain mult's recall and triples fusion's. We also ran the zero-shot accuracy suite on these arms, and at 63M and 10k steps it does not resolve. Every task sits within a point or two of chance and the spread between two mult-res seeds (39.9 against 39.1 on Avg_6) equals the spread across operators, so the choice rests on the two metrics that moved, loss and recall, and was confirmed at 340M where the mult-res model beats every linear arm on loss and both composites (Table 15).

D.4 Key and value modulation

Tiny Stories. The ablation proving that each mod helps on its own is run on TinyStories with the GPT-2 tokenizer, packed 512-token sequences and a 300M-token budget (1,144 steps of 512×512 tokens), well under one epoch, on an identical token stream for every arm. The trunk is an 8-layer pre-norm Transformer with RMSNorm, RoPE, SwiGLU and tied embeddings. The width-matched Transformer++ has d=512 and 8 heads (50.6M parameters). DM adds dual modulation to every layer of that trunk with the linear write operator, convex sigmoid gates reading $[\hat{\boldsymbol{x}}; \boldsymbol{u}]$, gate logits initialized at 0, no reserved raw head and no low-rank factoring, and is trained with the exact sequential scan (63.2M). The k-mod-only and v-mod-only arms drop one modulation (56.9M each). The parameter-matched Transformer++ widens the trunk to d=576 and 9 heads (60.8M), the closest head-multiple width to DM's count. All arms use AdamW$(0.9, 0.95)$, weight decay 0.1, learning rate 3×10^{-4} cosine to 3×10^{-5} with 1% warmup, gradient clipping at 1.0 and bf16, on 2× B300.

Fig. 4 and Table 17 give the result. Either modulation alone improves on the width-matched trunk, v-mod by 0.035 and k-mod by 0.040 nats, and both together improve on it by 0.058, which also beats the parameter-matched Transformer despite it having the wider trunk. The ordering is stable from about 100M tokens on, and the parameter-matched Transformer's two additional seeds both land above its primary run, so the margin holds on every seed run. The gains of the two mods overlap rather than add, which is what the k-mod head ablation (Table 13) would predict, since part of what each path buys is the same recurrence. The gate initialization matters. The same five arms run with gate logits at 4 (gate value 0.98, a near-vanilla start) give the opposite ordering, with the parameter-matched Transformer ahead of DM, because the model then has to learn to open the recurrent path against a near-closed gate. That is the observation behind the method section's initialization choice.

Formal languages. On CVP the two mods separate by function. Dropping key modulation at inference from a formed one-layer solver gives chance at every depth, so in a formed solver the gate addressing is carried by the contextual keys. The value path alone suffices, more slowly. A lineage

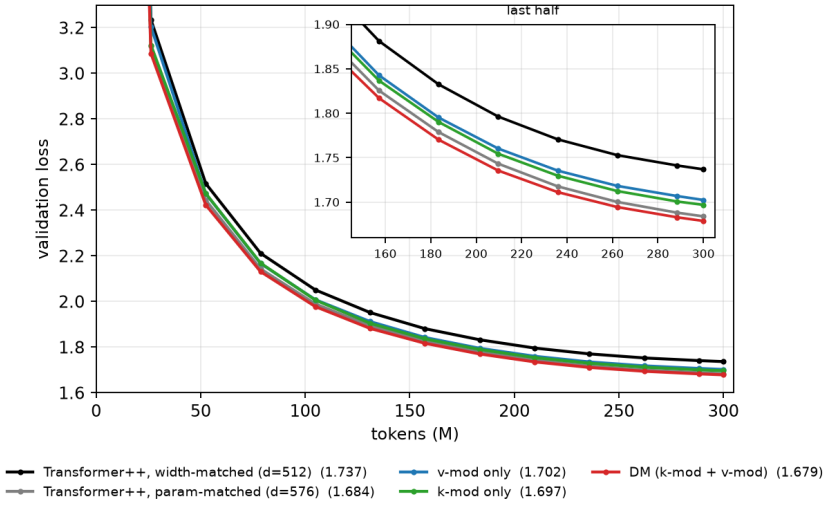


Figure 4: Validation loss on TinyStories over the 300M-token budget, evaluated every 100 steps, with the last half enlarged. Final values in the legend.

Arm	d	params	final val loss
Transformer++, width-matched	512	50.6M	1.737 (1.733, 1.734, 1.736)
v-mod only	512	56.9M	1.702
k-mod only	512	56.9M	1.697
Transformer++, param-matched	576	60.8M	1.684 (1.687, 1.693)
DM (k-mod + v-mod)	512	63.2M	**1.679**

Table 17: Final validation loss on TinyStories at 300M tokens. Additional seeds in parentheses where run.

trained with the key path off forms 16 gates in 3/5 seeds given about 80k steps, against 2/3 seeds by 60k with it on, and chains to 0.995 at 32 gates and 0.97 at 64. A formed solver with the key path deleted and fine-tuned recovers depths 1–8 within 3k steps and ends at 0.998, so the value path can re-learn the addressing from a formed model. The key modulation accelerates formation and carries addressing once formed, and the value modulation carries the composition, which is the division of labor the write-operator sweep (App. D.3) finds from the other side, where the key path carries no extra non-linearity and the value must, and which App. F argues is what separates FBT's depth tracking from its keyed reach.

E Length Generalization

Yang et al. (2026) characterize the regular languages on which Transformers length-generalize: exactly those in C-RASP, whose syntactic monoids divide an iterated wreath product of $\mathbb{Z}$. The two Krohn–Rhodes building blocks, flip-flops and finite groups, fall outside it, so parity, every group word problem and even "last symbol is b" are predicted to fail past the training range. We reproduce their protocol exactly: DFA state prediction with the state as a target only, training lengths sampled in $[2, 50]$, evaluation in bins out to 500 ($10\times$), 63M width, lr 4×10^{-4}, both RoPE and NoPE, with the separator convention for the flip-flop so that the last symbol is reachable only through attention. Their characterization is stated for models without positional encodings, so the NoPE columns are the direct test and the RoPE columns are the setting the rest of the paper uses.

Table 18 and Fig. 5 give the result. We see that every arm is flat at 1.0 on the two C-RASP languages, and the Transformer decays on every language outside it under both position encodings, reaching chance (0.54) on parity by $10\times$ and never fitting A_5 even in-distribution. The LSTM, a sequential model with the right inductive bias, is flat on all of them. DM sits between the two. On parity, two

Language	class	DM RoPE	DM NoPE	Transformer++ RoPE	Transformer++ NoPE	LSTM	FBT	Looped
$\Sigma^* a \Sigma^*$	C-RASP	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$\Sigma^* ab \Sigma^*$	C-RASP	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$\mathbb{Z}_2$ parity	group	**0.98** / 0.59	0.60	0.54	0.52	1.00	1.00 / 0.66	0.60 / 0.56
+ log-n logit scale	group	**0.98** / 0.75	—	0.56	—	—	—	—
$\mathbb{Z}_5$	group	0.34 / 0.29	0.32	0.27	0.32	1.00	1.00	0.30 / 0.29
$\mathbb{Z}_{10}$	group	0.26	—	0.15	—	1.00	0.68	0.20 / 0.19
A_5	group, NC^1	0.21 / —	—	—	—	0.985	1.00	0.19 / 0.08

Table 18: State-prediction accuracy at length 500 ($10\times$ the training maximum) on the protocol of Yang et al. (2026). The first two rows are in C-RASP, the rest outside. Paired cells are best / median for DM on parity (9 seeds, 2 at or above 0.8 at $10\times$; 7 seeds with the log-n scale, 3 at or above 0.8) and best / worst elsewhere (two DM seeds on $\mathbb{Z}_5$ and A_5, three FBT seeds on parity, the three loop recipes at their best learning rate and iteration count). "—" is a run that did not fit the training range (in-distribution accuracy ≤ 0.06) or was not run. The Transformer fits $\mathbb{Z}_{10}$ in range only to 0.57. On A_5 the looped pair is recurrent-depth / iteration-agnostic, and the instance-tied loop did not fit the range. "log-n" multiplies attention logits by $\ln n / \ln 50$ (Chiang and Cholak, 2022). The flip-flop $\Sigma^* b$ is omitted: under the separator protocol with relative positions it is a one-token copy and every architecture that finds it, including the Transformer, is flat to $32\times$.

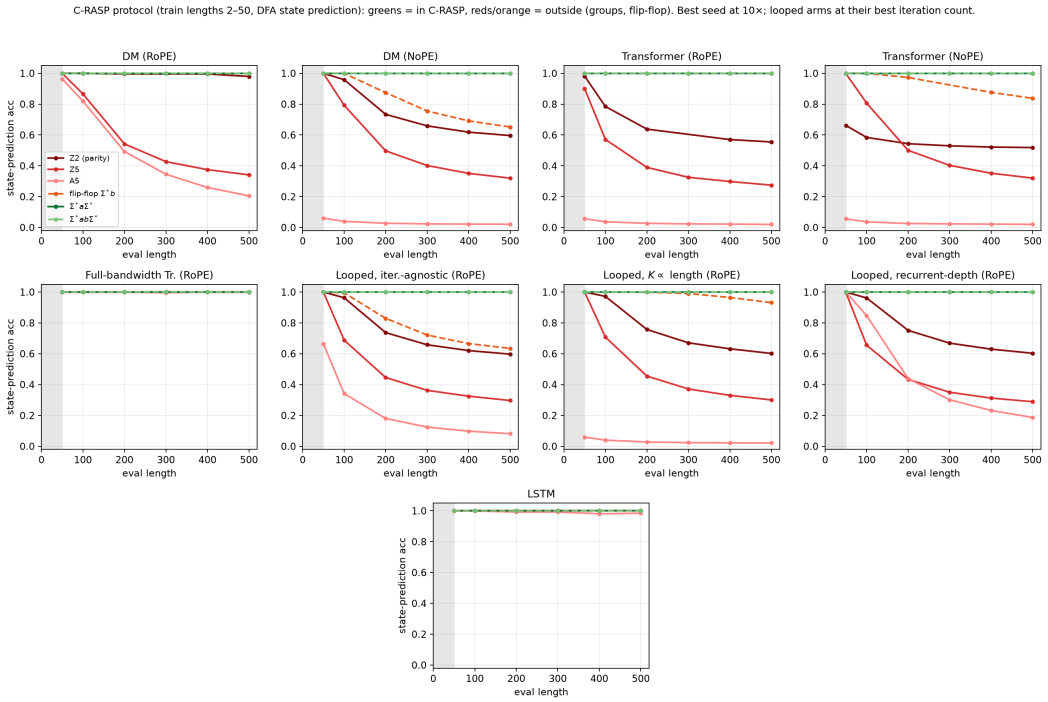


Figure 5: Length generalization on the C-RASP protocol (train lengths 2–50, shaded). Greens: languages in C-RASP ($\Sigma^* ab \Sigma^*$ dashed to separate it from $\Sigma^* a \Sigma^*$); reds and orange: outside (flip-flop dashed). Both Transformers and all three loops decay on every group; the LSTM and the full-bandwidth transformer are flat; DM holds when it forms the length-invariant solution and otherwise decays like a Transformer. Best seed per cell; looped arms at their best iteration count.

seeds in nine form a length-generalizing solution, the best of which holds 1.0 to $4\times$, 0.99–1.0 at $8\times$ and 0.98 at $10\times$, then decays slowly (0.91 at $16\times$, 0.86 at $20\times$, 0.70 at $32\times$); the other seeds hold 0.87–1.0 at $2\times$ and then decay from the training boundary like the Transformer (median 0.59 at $10\times$). On A_5 DM fits the task in one of three seeds and its solution decays out of range (0.21 at $10\times$), consistent with the $2 \times /4\times$ decay in Table 2. The full-bandwidth transformer is flat on

parity in two of three seeds (the third decays to 0.66 at $10\times$ and 0.55 at $32\times$), flat on $\mathbb{Z}_5$ and A_5 like the LSTM, and decays on $\mathbb{Z}_{10}$ (0.68 at $10\times$). The three looped recipes fit parity in every run and decay like the Transformer, to 0.56–0.60 at $10\times$ and 0.52–0.53 at $32\times$, decay on $\mathbb{Z}_5$ and $\mathbb{Z}_{10}$ to 0.29–0.30 and 0.19–0.20, and only the recurrent-depth loop fits A_5 in range before decaying to 0.19. Increasing the iterations beyond training at test time changes nothing. Looping therefore does not move the Transformer's length-generalization boundary, while both token recurrences cross it.

We inspect DM's decay mechanism. Reading attention mass on the formed parity solver, the relay is a broad read rather than a sharp hop to the predecessor (mass on key $t-1$ is ≈ 0.03 at every position and layer), and its peak dilutes with length, from 0.36 inside the training range to 0.25 at position 1000. Accuracy also collapses at the same peak-mass threshold (≈ 0.28) whether the sequence is 500 or 1000 long. This is therefore softmax dilution of a read whose logit gap does not grow with n. We find that scaling attention logits by $\ln n$ as suggested by Chiang and Cholak (2022), does not sharpen the read. Applied at inference it changes nothing, but training with it raises the typical seed (median 0.75 at $10\times$, $3/7$ seeds above 0.8) without moving the ceiling (best seeds identical with and without it).

F Looped and FBT baselines

Recipes. The iteration-agnostic loop iterates two (or nine) unique blocks K times with the input embedding re-added at every iteration and a learned step embedding per iteration (Bansal et al., 2022; Fan et al., 2025), trained at $K \sim U[K/4, K]$ per step with the same target at every K, and evaluated at every K from 8 to 128. The instance-tied loop uses the same block with K set to a quarter of the sequence length at training and at evaluation, so K scales with the length at extrapolation (Fan et al., 2025). The recurrent-depth loop follows Geiping et al. (2025). A prelude block embeds the input once, the recurrent state is initialized from a truncated normal of variance $2/5$, the recurrent block reads a linear adapter of $[\mathbf{s}; \mathbf{e}]$ with sandwich RMSNorm and no step embedding, one coda block precedes the head, the iteration count per step is $1 + \mathrm{Poisson}(\exp \mathcal{N}(\ln 32 - \frac{1}{8}, \frac{1}{2}))$, and backpropagation runs through the last eight iterations. Our shape is one prelude, two or nine recurrent and one coda block against their (2,4,2), and we use the harness learning-rate schedule. All three are reported at their best learning rate in $\{4, 2, 1\} \times 10^{-4}$.

Positive control. On a running-parity task in an interleaved format, each bit followed by a masked state slot whose target is the prefix parity (train 32 bits, test to 128), the looped transformer scores 1.0 at $4\times$ under both training regimes against 0.73 for the single-pass Transformer. This is the parity length-generalization gain that Fan et al. (2025) report, though on their own task, which supervises only the final parity end to end with the loop count tied to length; ours checks that the harness reproduces the gain, not that task. The interleaved format places the prefix states in the stream, which the tagging protocol removes.

Fixed-point measurement. For a trained loop, $\mathbf{x}_k$ is the recurrent state fed to iteration $k{+}1$, and the relative change is $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|/\|\mathbf{x}_k\|$ averaged over a batch. States are bf16, so a plateau near $2^{-8} \approx 3.9 \times 10^{-3}$ is consistent with element-level flicker at the last mantissa bit. Each trace is taken on in-distribution input and on out-of-distribution input at $2\times$ the length or $2\times$ the depth. Table 19 gives the values behind Fig. 3, and Table 20 the accuracy on the deepest bin at the same iterates, which is the number behind "further iterations change nothing."

Instance-tied loop on A_5. At $T{=}64$ the cold run at 4×10^{-4} reaches 0.24 in 12k steps. Continued at 2×10^{-4} and then twice at 5×10^{-5} it reaches 0.981 at 102k steps, and a further continuation reaches 0.9999 in-distribution, 0.78 at $2\times$ and 0.40 at $4\times$ at 132k steps, eleven times the budget at which DM and the LSTM form. A cold run at 10^{-4} needs none of this: 0.97 at 12k steps, 0.995 at 15k and 1.0 from 19k, with 0.60 at $2\times$ and 0.31 at $4\times$, which is the row in Table 2. On CVP the instance-tied loop forms 16 gates and sits at the shortcut floor at 32 in both seeds at every learning rate tried.

Reach. On CVP the iteration-agnostic loop is exact to 64 gates through a warm chain and reaches deepest exact depth 29 at 128 gates (0.72). The recurrent-depth loop is exact at 128 gates when evaluated with 64 or more iterations and walls at 256 in two seeds (0.78 and 0.83, exact to depth

Task	Recipe	k=2	16	32	64	128
A_5, T=64	iteration-agnostic	0.466	0.080	0.034	0.016	0.008
	instance-tied	0.292	0.283	0.054	0.020	0.009
	recurrent-depth	0.863	0.050	0.0054	0.0054	0.0053
keyed A_5, K=8, D=8	iteration-agnostic	0.431	0.048	0.024	0.012	0.006
	instance-tied	0.643	0.072	0.034	0.016	0.008
	recurrent-depth	0.701	0.030	0.017	0.017	0.017
CVP, 64 gates	iteration-agnostic	0.295	0.178	0.071	0.021	0.009
	instance-tied	0.316	0.055	0.028	0.015	0.008
	recurrent-depth	0.483	0.043	0.0041	0.0041	0.0041

Table 19: Relative state change per iteration, in-distribution input. The keyed iteration-agnostic and instance-tied runs and the CVP instance-tied run did not form and settle at a shortcut.

		in-distribution				2× length or depth			
Task	Recipe	k=16	32	64	128	k=16	32	64	128
A_5	iteration-agnostic	1.000	1.000	1.000	0.998	0.021	0.064	0.051	0.045
	instance-tied	0.878	0.999	0.998	0.998	0.016	0.018	0.018	0.018
	recurrent-depth	0.999	1.000	1.000	1.000	0.017	0.026	0.026	0.024
keyed A_5	recurrent-depth	0.990	0.990	0.991	0.991	0.158	0.708	0.716	0.714
CVP	iteration-agnostic	0.500	1.000	1.000	1.000	0.500	0.500	0.504	0.500
	recurrent-depth	0.840	1.000	1.000	1.000	0.523	0.523	0.512	0.516

Table 20: Accuracy on the deepest bin (positions 57–64, the deepest gate, or depth-D registers) at each iterate, for the formed runs. Chance is 0.017 on A_5 and keyed A_5 and 0.5 on CVP.

134 and 12), at every iteration count from 64 to 512. With one block it is exact at 16, 32, 64 and 128 gates and walls at 256 (0.75, exact to depth 3). On keyed A_5 the iteration-agnostic loop sits at depth 1 at every K and both learning rates, and the instance-tied loop holds depth 8 at K=2 and depth 1 at K=8. The recurrent-depth loop with two blocks holds depth 8 at K=8 through 256, chained by the register ramp, does not form at 512 at the batch size the memory allows, and with one block holds none at 23M or 62M at any learning rate.

Full-bandwidth transformer. Following Wang et al. (2026), Eq. 4, 8 and 13, the input to layer 0 at position t is $\mathbf{u}_t = \mathrm{RMSNorm}(\mathbf{W}^U \mathbf{h}_{t-1} \odot \sigma(\mathbf{W}^G \,\mathrm{RMSNorm}(\mathbf{e}_t)))$ with $\mathbf{u}_1 = \mathbf{e}_1$, where $\mathbf{h}_{t-1}$ is the previous position's final hidden state, $\mathbf{W}^G$ has no bias, $\mathbf{W}^U$ uses the standard initialization, and training adds jitter of 0.02 to $\mathbf{u}_t$. All FBT results in the paper's tables train the exact sequential recurrence, one teacher-forced pass with a KV cache and full backpropagation through every position with no truncation, unlike the few-pass approximation Wang et al. (2026) train at scale. FBT has no iteration axis, so it has no fixed-point trace.

Table 21 gives the training cost of the exact recurrence under the chunk scheme of §3.2 at the 1.3B schedule. A DM sweep recomputes the attention read and the low-rank contextual writes with the trunk MLP computed once, about a tenth of a layer, and its sweeps pipeline across layers and chunks. An FBT sweep recomputes the whole stack, since layer 0 at sweep k+1 needs the top layer at sweep k, so the critical path per chunk is K full stacks in place of one. The comparison assumes FBT's correctness front advances as fast as DM's 16 positions per sweep.

The few-pass schedule of Wang et al. (2026) is cheaper because it trains the model to be stable under a few parallel passes, which is the loop regime measured above. Pass 1 is a plain stack on the embeddings, each later pass re-runs the full stack in parallel on the previous pass's hidden states shifted one position and fused through the gate, gradients flow through every pass, and batches are 75% one-pass, 22% two-pass and 3% three-pass (their §3.3). Table 22 sweeps the pass count at the exact-trained arm's sizes and learning rates, decoded with the sequential unroll the recipe prescribes. Reach grows with the pass count, about two positions per pass on A_5, and the first task to form outright, 16 gates of CVP, needs $16\times$ the parallel FLOPs. Nothing forms at or below DM's cost. Read out with the training-time number of parallel passes instead of the unroll, the 3-pass

model scores 1.00 on keyed A_5 and the 8-pass model 0.91 on CVP-16, because the passes act as extra depth on the shifted states rather than as the recurrence, and the sequential decode does not reproduce that computation until the pass count approaches the sequence length. The pass count caps the reach the way the iteration count caps a loop, and the exact recurrence that solves all three tasks is the sequential one.

	f	FLOPs, $K{=}24$	FLOPs, $K{=}c{=}64$	stacks per chunk
DM	0.1	3.3×	7.4×	1
FBT	1.0	25×	65×	24

Table 21: Training cost under the chunk scheme of §3.2 at the 1.3B schedule ($c{=}64$), at the training operator ($K{=}24$) and at the exact chunk ($K{=}c$). f is the fraction of a layer recomputed per sweep, FLOPs are multiples of one parallel pass of the parameter-matched Transformer (DM's 3.3× is measured, the rest follow from $1 + Kf$), and the last column is the critical path per chunk in full stacks, since DM's sweeps pipeline across layers and FBT's cannot.

			A_5, $T{=}64$		CVP-16	keyed $K{=}8$
Passes P	FLOPs	serial steps	acc.	exact to pos.	acc.	depth
1.3 (paper)	1.3×	1–3	0.33 / 0.30	16	0.58	1
3	3×	3	0.10	8	0.54	0
4	4×	4	0.45	16	0.55	1
8	8×	8	0.62	28	0.57	6
16	16×	16	0.60	32	0.998	2
exact	1×	T	1.00	64	1.00	8

Table 22: The multi-pass FBT recipe against its pass count, at the sizes and learning rates of the exact-trained arm (12k steps on A_5 and keyed A_5, 30k on CVP), evaluated with the sequential unroll the recipe prescribes. FLOPs are multiples of one parallel pass, and serial steps are the number of full-stack applications per sequence that cannot run in parallel. "Paper" is the 75/22/3 mix of one-, two- and three-pass batches from Wang et al. (2026). The exact row is the sequential recurrence used in the paper's tables.

Serial composition against keyed addressing. FBT tracks CVP depth from one layer as DM does but holds keyed state only with depth: the one-layer 8M model solves $K{=}2$ and fails $K{=}8$ at depth 1 in two seeds, at two learning rates and also when widened to $d{=}1280$ (23M), while nine layers and 180M hold $K{=}2$, 8 and 16 (Table 3). We read this as a difference in where the recurrence enters. FBT's carry is the previous position's final state, so its token recurrence composes along adjacent positions. On CVP the chain $c = k - 1$ is adjacent and the three random operands are already-composed cache entries one lookup away, which suits it. On keyed A_5 a register's predecessor is not adjacent, so its state must be fetched by attention and composed within the stack at that position, and the entry is then written under a key derived from the adjacent carry and the raw token rather than from the retrieved state. DM composes at the read, where $\boldsymbol{u}_t$ is the retrieved register, and writes both the composed value and a key derived from it (Eq. 3), so a register's slot stays addressable across interleaving with one layer. The key-path ablation on CVP (App. D.4) points the same way: with the contextual key removed, DM still tracks depth from the value path alone but forms more slowly and in fewer seeds, and a formed solver loses its gate addressing when the key path is dropped. This is a mechanism inferred from the two tasks' predecessor structure and from the ablation, not one isolated in FBT. A keyed run of FBT with a key derived from the retrieved state is in progress to test this hypothesis.

T^{2}MLR (Cai et al., 2026) is FBT with the carried loop shortened to a middle block. On a stack that contains the block, an FBT can route the carry through the extra layers on the residual stream, so up to the fusion gate every T^{2}MLR is realizable by an FBT of the same depth, and our one-layer FBT, whose block is the whole stack, is also a one-layer T^{2}MLR. The adjacency argument above applies to it unchanged.

G ADDITIONAL STATE-TRACKING RESULTS

A_5 minimum depth by length. Fig. 6 gives the full length-by-depth grid behind the right half of Table 2, at fixed width d=768 and 12k steps. DM solves every cell with one layer, 45 of 48 seeds forming, and the three that did not are the cold-formation lottery rather than a depth effect, since deeper cells at the same length form in every seed. The Transformer's frontier grows with length. Under the strict rule that every seed must reach 0.95, it needs 3 layers at T=5, 8 at T=10 (4 and 6 layers form in one of two seeds), 12 at T=20, and no depth to 12 solves T=15 in every seed (best seed 0.89, 0/3). Table 2 uses the any-seed minimum, 2, 3, 6 and 4, so the two rules bracket the same picture: attention's depth requirement rises with the length, and the recurrence's does not.

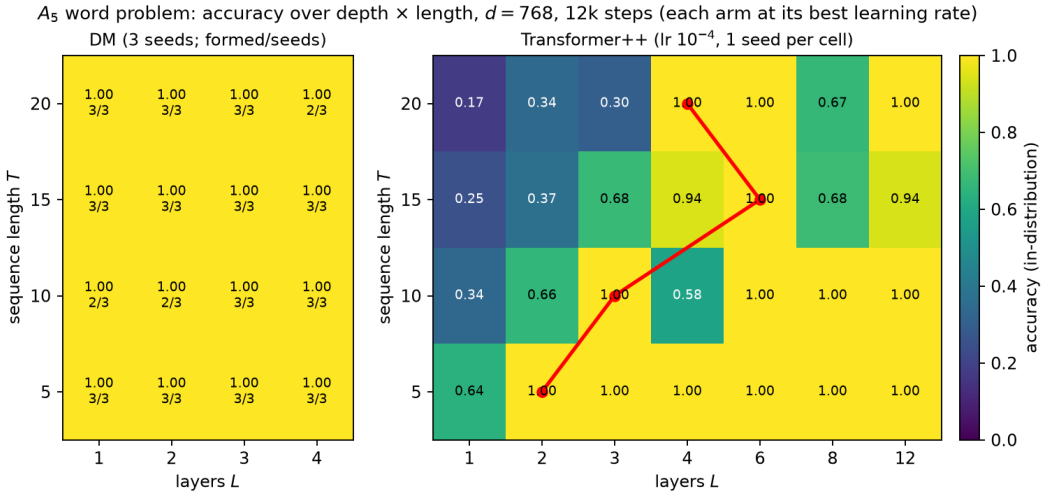


Figure 6: A_5 accuracy over sequence length × depth at fixed width d=768, 12k steps; formed/seeds printed where more than one seed was run. DM (left, 3 seeds per cell) solves every cell at one layer; 45/48 seeds form. The Transformer (right, best of 1–3 seeds) needs depth that grows with length: the red line is the minimum depth at which every seed reaches 0.95 (3 at T=5; 4, 6 and 8 at T=10, where 4 and 6 layers formed in one of two seeds; 12 at T=20), dotted where it crosses the T=15 row, which no depth to 12 solves in every seed (best seed 0.89, 0/3). The any-seed minimum used in Table 2 is 2, 3, 6 and 4.

CVP minimum layers. Fig. 7 gives the minimum number of layers for an exact solution against circuit size, sweeping layers at d=768. The LSTM needs two layers at 16 gates and no depth from 1 to 16 solves 32. Its exact reach stays at 16–28 for every depth from 2 to 8, and 16 layers do not train, so its wall is its state and not its depth. GDN needs nine layers at 16 gates, and no depth to 18 solves 32 (18 layers are exact to depth 24). Each doubling of its depth buys a few gates of reach, 7, 12, 25 and 28 at 2, 4, 9 and 18 layers on 64-gate circuits. The Transformer never leaves its layer-count reach. The loops are plotted by unique blocks: the recurrent-depth loop is exact to 128 gates with one block and the iteration-agnostic loop to 64, and neither with two blocks reaches the next size, so their reach is set by the iteration count rather than by depth. The one-layer DM is exact at every size. Fig. 8 gives the per-depth accuracy curves of every arm at 64 gates.

Extra DM layers do not help. Two- and nine-layer DM models seeded from the one-layer solver, with the extra blocks initialized to the identity, end at 0.91 and 0.89 at 64 gates, below the one-layer chain's 1.00, and cold nine-layer models park at 0.55–0.71. On this task the recurrence over the tape carries the computation and extra depth is a liability for formation.

The 1024-gate cell. Three lineages, each formed cold at 16 gates, reach 256 gates exact. Two reach 512. One reaches 1024, where the 60k budget is exact to depth about 780 and a 26k-step continuation at 2×10^{-5} completes it with minimum per-depth accuracy 0.984. The second lineage at 1024 is exact through depth 794 at the time of writing. The learning-rate drop matters. Continuations at the

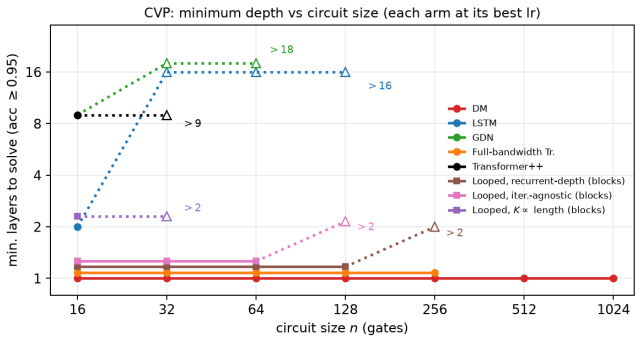


Figure 7: CVP, minimum number of layers for an exact solution (accuracy ≥ 0.95 at every depth, d=768) against circuit size. Solved cells at 16 gates and every unsolved depth sweep are cold 30k-step runs; solved cells beyond 16 gates (DM, FBT, looped) are warm chains from the previous size. Open triangles mark sizes where no depth up to the one marked reached 0.95. Looped arms (squares) are plotted by unique blocks, each applied K times per token, so their height is not a depth.

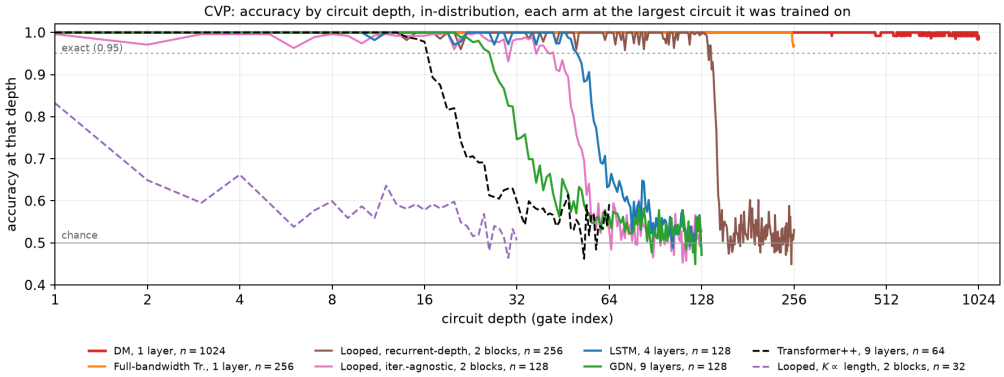


Figure 8: CVP accuracy by circuit depth, in-distribution, each arm at its best learning rate and at the largest circuit it was trained on (DM and the full-bandwidth transformer one layer, LSTM four, GDN nine, Transformer++ nine, looped two blocks); dotted line is the exactness threshold, solid grey is chance.

original rate from a nearly formed solver reset the deep front, which is the same shock seen in the keyed register ramps.