

Is the graph doing anything?

A sourced survey of recurrent self-routing neural graphs as an alternative to layered networks

By **Pavle Lazić** · ScalablyAI · Built with AI agents, reviewed by AI models

How this was made. The survey, evidence ledger, analysis and this note were produced by Claude Opus 5.5 agents (Anthropic), directed by Pavle Lazić, and reviewed by fresh-context Claude Opus 5.5 reviewers and by GPT-6 Astra (OpenAI) as a second model family. It has not yet been reviewed by human experts. No experiments were run; every number below is a value reported by a cited source or computed from one.

Code, data and protocol: <https://github.com/scalably-io/neural-graph-substrate> · doi:10.5281/zenodo.23091812

1. Summary

Question. Can a persistent recurrent graph of neural nodes, which shares one message and one update function, rewires itself from its own state at every step, and computes by evolving until it halts, give more reasoning capacity per stored parameter than a Transformer, because the same weights are reused in different graph configurations?

Method. A quote-per-claim evidence ledger of 962 citable claims from 6 research workstreams across twenty neural-network families, a 121-row matrix answering fourteen fixed questions per approach, 7 adversarial reviews by fresh-context AI reviewers from two model families, which were allowed to reverse our conclusions, and a dedicated prior-art search in Transformer vocabulary.

Answers.

1. Against a fixed-depth Transformer, per stored parameter: recurrence helps (looped language models report 2 to 3 times parameter efficiency against other teams' models trained on different data, [R2-016](#); a controlled study puts looping a block r times at about r to the power 0.46 distinct blocks, [R4-185](#)), but no cited evidence tests whether the graph adds to that, and a weight-shared looped Transformer already has it.
2. Against a looped Transformer: open. The closest study (LT2) compares dense, sparse, recurrent and hybrid mixers inside the same loop; which one wins depends on the task (Table 2), and no comparison in it isolates routing.
3. One controlled study finds recurrence does not raise knowledge storage: about 2 bits per parameter, looped or not.
4. Most components of the design are published, but separately. In our scoring of 15 rows (one of them a union of looped-Transformer papers), no row fully has more than 7 of the 13 components, and none fully has C10, C13 (Figure 1). These are results of our scoring, not proofs of absence. The full combination was not found in our search.
5. What remains open is narrow. LT2 already compares looped top-k, recurrent and dense mixers on retrieval beyond the training length and, across loop counts, on a curriculum trained at each size. Still open is whether learned, unsupervised top-k routing over all nodes beats a sharpened looped Transformer and looped recurrent controls on instance-size extrapolation in algorithmic problems. We publish a protocol to test it (Section 7) rather than results.

What would change these answers. A matched experiment in which learned routing beats a dense looped Transformer, a sharpened one, and looped recurrent and hybrid controls on size extrapolation would revise answer 2; a published version of that experiment would close answer 5.

2. The question and its scope

The brief asked for graph-native architectures on ordinary digital hardware (GPU, TPU, CPU). The candidate has 13 components, labelled C1 to C13 in Figure 1: persistent node states, a sparse directed graph, a shared message function, a shared update function, recurrent evolution over many steps, learned top-k routing or edge creation, hyperedges, topology conditioned on the current state, persistent associative memory, readouts attached to many graph regions, a confidence or convergence signal, learned adaptive halting, and topology and node roles that evolve during computation.

We separate two kinds of capacity, because the evidence treats them differently. Computational capacity is the size or difficulty of problem a model can solve; recurrence can plausibly raise it. Knowledge storage is what a model memorises; one controlled study finds recurrence does not raise it (Section 4.3).

We added one comparator the brief did not list: the weight-shared looped Transformer. Self-attention can be written as message passing on a complete token graph whose weights depend on the current state, as Joshi argues (R4-070, an author claim; R1-099 is the same paper), and a looped Transformer already reuses one set of weights over time. Beating a fixed-depth Transformer would answer the brief's performance question, but would not show whether routing adds anything beyond recurrence.

3. Method

Evidence ledger. Every row holds one claim, the source URL, a locator, a verbatim quote that contains the number when there is one, a status, and the date read. Quotes aim at forty words or fewer; the validator allows up to sixty, and 7 rows exceed forty. Rows enter the ledger only through a validator that rejects missing quotes, malformed IDs and reported numbers absent from their quote. The ledger holds 962 citable claims (358 reported results, 103 theorems, 174 design facts, 208 author claims, 107 repository facts, 12 of our own derivations) and 3 sources we could not access, which are kept but never cited. Author claims are attributed, never stated as fact.

Approach matrix. 121 approaches, each answering the brief's fourteen questions (computational graph, learned edges, topology at inference, persistent state, weight sharing, training, scaling, stability, capacity, whether more iterations help, GPU fit, repositories, best results, open problems). The matrix is only partly filled: for whether extra iterations help, 62 rows say unknown and 19 say not applicable.

Review. Three fresh-context Claude reviewers, not told our reasoning, attacked the survey and the experimental design and found 11 blocking errors; three more attacked drafts of this note; GPT-6 Astra then reviewed it as a second model family. Every finding was re-derived or re-fetched before we accepted it; the record is public (Section 8).

Prior-art search. The first workstreams searched mostly in graph-network vocabulary; the last searched in Transformer vocabulary and found the closest prior work (Section 5) that the earlier searches had missed. Rows added by the orchestrator while verifying reviewer findings are kept in their own file.

4. What the evidence says

4.1 Time can replace depth, and a looped Transformer already does it

Weight-tied recurrence extrapolates on algorithmic tasks when the input is fed back at every step and training does not tie behaviour to one iteration count: a recurrent network trained on 32-bit prefix sums reaches a peak of 97.12% on 512-bit strings, the best over test iterations, chosen after the fact (R4-024), and a recurrent graph network extrapolates to graphs 1000 times larger than in training (R1-027). Without a stabiliser it collapses after about 100 rounds (R1-029). Saunshi and colleagues report that k layers looped L times nearly match kL distinct layers (R4-080, an author claim). In the one parameter-matched test of a brain-inspired recurrent structure we found, a plain Transformer in the same recurrent pipeline came within about 5 points of it, without hyperparameter optimisation (R4-165): the hierarchical structure added at most that much, and an outer refinement loop drove most of the gain (R2-048).

4.2 Per parameter the gain is large; per unit of compute it is small or negative

Looped language models report 2 to 3 times parameter efficiency (R2-016), measured per stored parameter rather than per unit of compute; by our derivation the smaller looped model spends about 1.4 times the forward compute of its comparator, ignoring attention and early exit (R2-199). One study that loops the middle layers of a mixture-of-experts model, with parameters, compute per token and cache all matched, saves 6.8 to 18.0% of training compute (R2-089). An iso-depth study finds a looped model with 410M parameters at $r = 4$ matches a 580M model in loss but costs the training compute of a 1B model (R5-092), and looping a block r times is worth about r to the power 0.46 distinct blocks in language-model loss, where an exponent of one would mean full equivalence (R4-185). At language-model scale, extra test-time loops degrade after the trained depth of 4 (R2-018) or saturate (R2-095).

4.3 Recurrence does not add stored knowledge

A controlled study finds about 2 bits of knowledge per parameter for looped and non-looped models alike (R2-017). If that holds beyond looped Transformers, a recurrent graph would gain computation, not stored knowledge; the design's persistent associative memory (C9) is a separate question this evidence does not settle.

4.4 Routing: what is already published

Every routing ingredient of the design exists separately (Figure 1, Table 1).

- Per-query top-k attention dates from at least 2019 (R6-008). A 2025 model combines a recurrent reasoning unit with per-query top-k attention (R6-006, R6-007).
- A recurrent graph network with one weight-shared layer and dynamic pathways exists (R1-067), and per-node, state-conditioned roles (listen, broadcast, isolate) exist in a non-recurrent graph network (R1-059).
- Hard attention inside a recurrent processor is reported as important for size generalisation, tested on graphs of 16 to 1600 nodes (R6-010, an author claim; R6-011); it is trained with supervision on the algorithm's state transitions (R4-048, an author claim).
- The cited paper proves a dispersion limitation for softmax attention as the number of items grows (R6-001, read from the abstract). Sparser or sharper normalisations and selective attention mitigate it without hard top-k selection: one reports up to 1000 times length extrapolation on synthetic tasks (R6-016), and a weight-tied Transformer whose sharp, selective attention its authors describe as a form of neural routing reaches 100% length generalisation on a lookup task (R6-012).
- Restricting attention to the task's own edges helps, when supervised step by step: soft attention on the input graph reaches 99.97% last-step reachability at test size, against 91.51% for attention over the complete graph (92.34% against 88.98% averaged over steps), and max aggregation on the same edges reaches 99.80% (R6-017, R4-001, R4-002). The gap is graph structure, not discrete selection.
- Learned hyperedges exist (R1-109), and one recurrent model builds a new incidence matrix at every step from its current state and passes messages weighted by it, with parameters shared across steps (R1-164, R6-034, R6-035, R6-036): learned, state-conditioned hypergraph routing inside a loop is published, though without hard top-k selection.

Two results bear on whether routing, rather than something simpler, explains gains. First, pure locality: adding one causal convolution lifts a recursive model's state-tracking accuracy at length 128 from 45.8% to 91.4% (R7-032, R7-033). Second, relational attention with evolving edge states: a Transformer with them scores 81.30% against 42.34% without them on algorithm execution (R4-176). Neither is hard top-k routing, but the second is close to the brief's stateful relational computation.

Most parts exist somewhere. In our scoring, no system has more than 7 of 13.

The 13 components of the proposed recurrent self-routing graph (columns) in the 15 closest published systems (rows; the first is a union of looped-Transformer papers). Every filled or ringed mark rests on a cited claim whose quote supports it (hover for IDs). Fully present in no system: C10, C13.



Grey "does not" comes from the researchers' coverage tables; "?" means we found no supporting claim in our ledger (re-scored after reviews N1 to N3 and Astra, 2026-10-01).

scalably.io · neural-graph-substrate

Figure 1. The components of the proposed design (columns) in the closest published systems (rows). Filled and ringed marks rest on a cited claim (some of them author claims, flagged in `evidence/coverage.csv`); grey dots come from the researchers' coverage tables and are not individually sourced; a question mark means we found no supporting claim. The counts are results of this scoring, not proofs of absence.

Table 1. The data behind Figure 1 (claim IDs per cell are in `evidence/coverage.csv`).

SYSTEM	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Looped Transformer family (union of several papers)	not established	no	yes	yes	yes	no	no	not established	no	not established
LT2 looped sparse attention (2026)	not established	partly	yes	yes	yes	yes	not established	partly	not established	not established
ReSSFormer (2025)	yes	yes	yes	yes	yes	yes	not established	yes	partly	not established
Mixture-of-Recursions	not established	not established	not established	not established	yes	partly	no	partly	no	no
HRM / TRM recursive reasoning	not established	no	not established	yes	yes	no	no	no	no	not established
N2 recurrent graph network (2024)	yes	partly	yes	yes	yes	partly	no	partly	no	no
Co-GNN	partly	partly	partly	no	no	partly	no	yes	no	no
IterGNN	not established	not established	not established	not established	yes	no	no	no	no	no
Discrete NAR (2024)	not established	partly	not established	not established	yes	partly	not established	partly	not established	not established

SYSTEM	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Pointer Graph Networks / NEE	not established	partly	not established	not established	not established	partly	not established	partly	not established	not established
Continuous Thought Machine	yes	no	partly	no	yes	no	no	partly	no	partly
Energy Transformer	yes	no	yes	yes	yes	no	no	partly	yes	no
Graph Neural Cellular Automata	yes	yes	yes	yes	yes	no	no	no	no	no
Set-to-hypergraph recurrent refiner	yes	not established	yes	yes	yes	not established	yes	yes	not established	not established
DyHSL learned hypergraph	not established	not established	not established	not established	not established	not established	yes	not established	not established	not established

4.5 On GPUs, an efficiency gain at these sizes is unshown

At the sizes algorithmic benchmarks use, we infer (without a measurement at these sizes) that dense masked attention is the fastest implementation: gather and scatter message passing is memory-bound in large-graph measurements (R5-086), and randomly placed top-k edges leave essentially no empty tile for block-sparse kernels to skip (R5-131, our derivation under stated assumptions). PyTorch's built-in FLOP counter counts nothing for scatter, gather or top-k (R5-049). At long context, block- or page-structured top-k attention does pay (R5-017, R5-041, R5-042); unstructured per-node top-k, as in this design, is not shown to. Our hypothesis, not a result: at these sizes the case for the design rests on inductive bias rather than efficiency, unless a measurement with learned, local edge patterns shows otherwise (the tile calculation assumes random targets).

5. The closest prior work

LT2 (arXiv 2605.20670) compares dense, sparse (learned top-k), recurrent and hybrid mixers inside the same weight-tied loop, at what the authors describe as the same parameter budget (R7-007, an author claim). We report its long-context results in full (Table 2 and Table 3). The source states its model size and hybrid ratio inconsistently (the caption and the table header give different model sizes, and the setup and the caption give different ratios of mixer to attention layers; R6-037, R6-038, R6-039), so we rely only on its reported scores.

- **A curriculum trained at each size**, sweeping the loop count (R7-004): the dense loop's best is a largest solved size of 64, while four other mixers reach 128 at their best loop count (R7-005, R7-006, R6-028); two of those four have no routing, and one of them reaches it only at eight loops (R6-032). The authors read this as looping helping cheaper mixers more than full attention (R6-027, an author claim).
- **Language-model evaluation** (Table 2 and Table 3): on most knowledge benchmarks the dense loop scores at or near the top, though not on all of them, and the authors present the routed hybrid as tracking it closely (R7-012, an author claim); on retrieval beyond the training length, dense attention scores zero, looped or not, while recurrent and hybrid mixers do not, and the authors attribute extrapolation to recurrent backbones (R6-026, an author claim).

Table 2. LT2's knowledge benchmarks at the training length, every model row and every column (R6-018 to R6-025).

MODEL (LOOPED = WEIGHTS SHARED OVER FOUR STEPS)	MIXER	SWDE	SQUAD	FDA	TQA	NQ	DROP	CLAIM
Transformer	dense attention	48.9	46.6	58.4	67.5	31.7	26.4	R6-018
GDN	gated linear recurrence (no routing)	32.7	40.0	28.3	63.5	25.7	24.5	R6-019
Mamba-2	state-space recurrence (no routing)	30.7	39.1	23.7	64.3	25.1	28.5	R6-020

MODEL (LOOPED = WEIGHTS SHARED OVER FOUR STEPS)	MIXER	SWDE	SQUAD	FDA	TQA	NQ	DROP	CLAIM
Looped Transformer	dense attention	52.8	49.4	61.7	68.2	33.6	28.1	R6-021
Looped GDN	gated linear recurrence (no routing)	34.9	41.8	30.6	64.7	27.0	25.9	R6-022
Looped Mamba-2	state-space recurrence (no routing)	33.9	40.5	25.8	65.1	26.8	29.7	R6-023
Looped Hybrid (GDN+DSA)	linear recurrence + learned sparse top-k attention (routing)	51.6	48.0	60.4	66.9	33.0	28.4	R6-024
Looped Hybrid (Full+GDN)	full attention + linear recurrence (no routing)	53.1	48.9	62.0	67.8	34.0	30.2	R6-025

Table 3. LT2's retrieval tests at three lengths, every model row ([R6-018](#) to [R6-025](#)); models were trained at 2048 tokens, so the 4096 columns test extrapolation.

MODEL	TEST 1, SHORTER	TEST 1, TRAINED LENGTH	TEST 1, LONGER	TEST 2, SHORTER	TEST 2, TRAINED	TEST 2, LONGER	TEST 3, SHORTER	TEST 3, TRAINED	TEST 3, LONGER	CLAIM
Transformer	100.0	100.0	0.0	92.2	100.0	0.0	98.6	99.4	0.0	R6-018
GDN	100.0	100.0	99.8	100.0	93.8	49.8	83.8	68.4	34.2	R6-019
Mamba-2	100.0	99.6	62.0	100.0	53.8	11.8	95.8	87.4	13.4	R6-020
Looped Transformer	100.0	100.0	0.0	94.6	100.0	0.0	99.2	99.8	0.0	R6-021
Looped GDN	100.0	100.0	99.8	100.0	96.4	53.2	85.6	71.0	35.8	R6-022
Looped Mamba-2	100.0	100.0	65.7	100.0	57.1	13.5	96.2	88.1	16.2	R6-023
Looped Hybrid (GDN+DSA)	100.0	100.0	91.4	100.0	100.0	77.6	100.0	99.6	60.3	R6-024
Looped Hybrid (Full+GDN)	100.0	100.0	93.5	100.0	100.0	81.0	99.8	99.8	63.7	R6-025

Which mixer is best depends on the task, and none of LT2's comparisons varies routing alone. We found LT2 only when we searched in Transformer vocabulary. A study of the open question would extend it, and should include looped recurrent and hybrid mixers as controls alongside the dense and sharpened ones.

6. Novelty map

- **Established:** attention as soft, state-dependent message passing; softmax dispersion as the number of items grows, and sharper attention functions that mitigate it; per-query top-k attention, and recurrence combined with it; time replacing depth on algorithmic tasks, with stabilisers.
- **Reported by one controlled study:** recurrence adds computation but not stored knowledge.
- **Partly explored:** state-conditioned per-node roles and topology, per layer; edge creation with step supervision; hard selection for size generalisation, supervised; learned, state-conditioned hypergraph routing inside a recurrent loop; cheaper mixers, including top-k sparse attention, compared with dense attention inside a loop (LT2).
- **Not found in our search:** learned, unsupervised top-k routing over all nodes inside a weight-tied loop, compared with a dense and a sharpened looped Transformer, looped recurrent and hybrid controls, and a fixed local graph, at equal steps and no more matrix-multiply compute per step, on train-small, test-large problems. Also not found: hard top-k selection over learned hyperedges, or evolving node roles, inside a recurrent loop.

7. A protocol for the open question

We publish, but did not run, a preregistration-ready protocol ([docs/PREREGISTRATION.md](#)), with code in [ngs/](#)).

- **Arms.** A fixed-depth Transformer; a looped Transformer; a looped Transformer with sharpened attention; a fixed-depth and a looped graph network on the task's own edges; the candidate, a looped model whose every node attends to its top 8 nodes by current state; and the candidate with learned halting. All looped arms share one training recipe and receive the task's edges as a learned score bias; they differ in the attention or routing operator, plus a halting head in the last arm. After LT2, looped recurrent and hybrid controls must be added (open item six of the protocol).
- **Tasks.** Six families with exact solvers and train-small, test-large ladders: reachability, shortest path, mazes with and without cycles, sorting, associative recall and state tracking.
- **Budget.** Every looped arm runs the same number of steps, derived from the task alone: 1.5 times the steps a one-hop exact algorithm needs at the ninety-ninth percentile (reachability: from 12 steps at 16 nodes to 56 steps at 1024 nodes; mazes: from 81 steps at 9 cells per side to 737 steps at 33 cells per side). Answers are read at the last step.
- **Matching.** Stored parameters within 5%; the candidate never executes more matrix-multiply compute per step than the dense loop.
- **Statistics.** Paired seeds, between 5 and 15 per arm from a power rule; four mutually exclusive outcomes per comparison (superior, inferior, equivalent within 0.05 of normalised area under the accuracy curve, inconclusive); comparisons where both arms are at floor or both at ceiling do not count.
- **Before freezing,** six design questions remain open and are listed in the protocol.

8. Negative findings, including our own errors

The project log records 42 reversals and errors, each with its evidence. The most consequential:

- Our first draft said the evidence leaned against the design; its citations compared looped with non-looped Transformers, not graph with looped. We now say the question is open.
- We first believed the mechanism we wanted to test (soft attention spreading thin as problems grow) was ours to test; it is a published theorem with cheaper fixes.
- We first claimed the architecture was a new combination; recurrence with top-k attention, recurrent graph networks with dynamic pathways, hard attention in a loop, and sparse versus dense attention in a loop are all published.
- Our first experimental design would have credited the candidate with cheaper per-step compute as if it were inductive bias, could have produced a null result from floor effects alone, and chose the stopping step using test labels. Reviewers caught each; the published protocol fixes them.

9. Limitations and the strongest case against this note

- No experiment was run. Every verdict about routing is about the literature, not about a model we trained.
- Our searches were bounded: one day, a fixed set of queries per workstream, and one search engine rate-limited for much of the prior-art sweep. "Not found in our search" is not "does not exist". One closely related paper could be read only as an abstract, and 14 of the claims this note cites use an abstract as their locator, including the dispersion theorem and the sharpened-attention results the protocol's main control relies on.
- Figure 1 is our scoring: its grey "does not" marks are not individually sourced, cells we could not support are marked not established, and a fuller reading could raise counts. A second-model review raised one system's count after reading its full text.
- The reviewers were AI models (Claude and GPT-6). Human experts in looped Transformers or neural algorithmic reasoning may weigh the evidence differently.
- The strongest case against us: in LT2 the dense looped Transformer, our main comparator, loses on retrieval beyond the training length and on its curriculum, so a recurrent graph could beat it for reasons unrelated to routing, and a reader could take that as support for the design. Equally, if routing survives the

sharpened-attention and recurrent controls on train-small, test-large problems, the design's core bet is right and this note is too cautious.

10. Reproduce and contribute

The ledger, matrix, coverage table, validator, task generators, step-budget and compute models, and the build of this note are in the project repository. Every number in this note is filled by `paper/build_paper.py`, never typed: reported results come from claim quotes (cross-checked against each claim's value), and the rest from claim text, our derivations, the coverage table, protocol constants or inventory counts. The build's checks are structural (it fails on a typed number, an unknown or inaccessible claim ID, a dash or a hype word); they do not verify meaning, which is what the reviews were for. To challenge a number, find its claim ID in Appendix A and check the quote against the source.

Code under Apache-2.0; text, data and figures under CC BY 4.0.

Appendix A. Evidence rows cited in this note

Each row is copied from the ledger ([evidence/claims.csv](#), 965 rows in all), with its verbatim quote. Status says what kind of evidence it is: a vendor statement is not a measurement.

ROW	CLAIM	VALUE	STATUS	SOURCE	VERBATIM QUOTE
R1-027	Recurrent GNNs (RecGRU-E) trained on size-10 graphs extrapolate to graphs 1,000 times larger	1000 x training size	REPORTED	arxiv.org · Experimental Evaluation, Table 1 caption	"The RecGRU-E outperforms IterGNN and can extrapolate well to graphs that are 1,000 times larger than the graphs encountered during training."
R1-029	Without L2 state regularization, accuracy declines rapidly after ~100 rounds; with it, stable up to 10,000 rounds (graphs of size 10)	100 rounds	REPORTED	arxiv.org · Section on stabilization, Figure 4	"both versions output correct predictions until approximately 100 layers. Afterwards, the models trained with L2 regularization still stay stable while accuracy declines rapidly without regularization."
R1-059	Co-GNN: each node chooses per layer to listen/broadcast/both/isolate via an action network given its state and neighbors		DESIGN	arxiv.org · Section 4	"an action network π predicts, for each node v , a probability (ℓ) distribution $p_v \in \mathbb{R}^4$ over the actions $\{S, L, B, I\}$ that v can take, given its state and the state of its neighbors N_v "
R1-067	N2: a single shared recurrent layer moves graph nodes and pseudo nodes in a common space, building dynamic pathways at linear complexity		DESIGN	arxiv.org · Abstract; Section 1	"N2 incorporates a recurrent layer to parameterize the displacements of graph nodes and pseudo nodes in the common space."
R1-099	Transformers are message-passing GNNs on fully connected token graphs that win the hardware lottery via dense matmuls		AUTHOR-CLAIM	arxiv.org · Abstract	"Transformers are implemented via dense matrix operations that are significantly more efficient on modern hardware than sparse message passing. This leads to the perspective that Transformers are GNNs currently winning the hardware lottery."
R1-109	DyHSL learns hypergraph structure by low-rank decomposition, contrasted with DHGNN kNN/K-means hyperedge construction		DESIGN	arxiv.org · Related work	"Compared to DHGNN that builds hypergraphs using kNN and K-Means algorithm to cluster node features, our DyHSL explicitly learns the structure of the hypergraph based on low rank matrix decomposition"
R1-164	Set-to-hypergraph model uses a recurrent network that iteratively refines edge and node features to predict the incidence matrix (learned hyperedges)		DESIGN	arxiv.org · Section 1 / Section 3	"We propose a simple recurrent neural network that refines the edge and node features, based on which it predicts the incidence matrix while preserving the permutation"
R2-016	Ouro LoopLM 1.4B and 2.6B trained on 7.7T tokens match 4B and 8B standard transformers, 2-3x parameter efficiency	2 x parameter efficiency (2-3x)	REPORTED	arxiv.org · §1 Contributions	"we demonstrate that 1.4B and 2.6B parameter LoopLMs match 4B and 8B standard transformers on most benchmarks, yielding 2-3x parameter-efficiency gains"

ROW	CLAIM	VALUE	STATUS	SOURCE	VERBATIM QUOTE
R2-017	Ouro controlled study: recurrence does not increase raw knowledge storage, ~2 bits/parameter for looped and non-looped	2 bits per parameter	REPORTED	arxiv.org · §1 Contributions (and §6.1)	"We find recurrence does not increase raw knowledge storage (approximately 2 bits per parameter for looped and non-looped models) but dramatically enhances knowledge manipulation capabilities"
R2-018	Ouro performance peaks at trained depth T=4 and degrades when extrapolating to T=5..8	4 recurrent steps (trained max)	REPORTED	arxiv.org · §5.3, Table 10	"Performance peaks at the trained depth (T = 4) and then degrades."
R2-048	ARC Prize: outer refinement loop drives performance: +13pp from 1 to 2 loops; 1->8 refinement loops doubles public eval score	13 percentage points	REPORTED	arcprize.org · Finding #2, Figure 4	"from no refinement (1 loop) to just 1 refinement, performance jumps by +13pp. From 1 to 8 refinement loops, the Public Evaluation set performance doubles."
R2-089	SMELT: looping MoE middle layers twice with per-token FLOPs, total params and KV cache all matched saves 6.8-18.0% training FLOPs on compute-optimal frontier	6.8 % training FLOPs saved (6.8-18.0)	REPORTED	arxiv.org · Abstract	"SMELT's loss drops faster with compute, saving 6.8–18.0% of training FLOPs on the compute-optimal frontier."
R2-095	Parcae: test-time looping follows a saturating exponential decay $L(T)=L_{\text{inf}} + Ze^{-zT}$		REPORTED	arxiv.org · §6 test-time scaling	"We find that the test-time scaling curves are well-described by a saturating exponential decay of the form: $L(T) = L_{\infty} + Ze^{-zT}$."
R2-199	Ouro 1.4B at 4 recurrent steps spends ~5.6B-dense-equivalent forward FLOPs/token, so matching a 4B dense model is a parameter win but not a FLOP win	5.6 billion-param-equivalent FLOPs/token	DERIVED	arxiv.org · Figure 1 caption + §1 contributions	"Radar plots comparing the Ouro 1.4B and 2.6B models, both with 4 recurrent steps (red), against individual transformer baselines."
R4-001	Neural execution of graph algorithms: an MPNN with max aggregator, trained on 20-node graphs, predicts reachability (BFS) at 100 nodes with 99.80% last-step accuracy.	99.80 % last-step accuracy, 100-node test	REPORTED	arxiv.org · Table 1 (Reachability), Sec. 4	"MPNN-max (Gilmer et al., 2017) 100.0% / 100.0% 100.0% / 100.0% 99.92% / 99.80%"
R4-002	Same table: fully-connected attention (GAT-full, labelled Vaswani et al. 2017) reaches only 91.51% last-step reachability accuracy at 100 nodes vs MPNN-max 99.80%, trained at 20 nodes.	91.51 % last-step accuracy, 100-node test	REPORTED	arxiv.org · Table 1	"GAT-full* (Vaswani et al., 2017) 78.40% / 77.86% 85.76% / 91.83% 88.98% / 91.51%"
R4-024	DT-Recall with progressive loss, trained on 32-bit prefix sums with at most 30 recurrent iterations, solves 512-bit strings at 97.12% peak accuracy (237 test iterations); plain DT gets 11.26%, FF 0.00%.	97.12 % peak accuracy, 512-bit test	REPORTED	arxiv.org · Appendix A.5, Table 3 (Tested on 512-bit Strings)	"DT-Recall 1.0 237 97.12 ± 1.88"
R4-048	Discrete NAR forces execution through finite predefined discrete states; with state-transition supervision it achieves perfect test scores and provable correctness for any test size.		AUTHOR-CLAIM	arxiv.org · Abstract	"Trained with supervision on the algorithm's state transitions, such models are able to perfectly align with the original algorithm. To show this, we evaluate our approach on multiple algorithmic problems and achieve perfect test scores"
R4-070	Joshi 2025: Transformers are message-passing GNNs on fully connected token graphs, with attention giving relative importance of all tokens.		AUTHOR-CLAIM	arxiv.org · Abstract	"We show how Transformers can be viewed as message passing GNNs operating on fully connected graphs of tokens, where the self-attention mechanism capture the relative importance of all tokens w.r.t. each-other"
R4-080	Saunshi et al. 2025: a k-layer Transformer looped L times nearly matches a kL-layer non-looped model on addition, p-hop induction and math, and beats a k-layer model.		AUTHOR-CLAIM	arxiv.org · Abstract	"a k-layer transformer looped L times nearly matches the performance of a kL-layer non-looped model, and is significantly better than a k-layer model."
R4-165	ARC Prize analysis of HRM: a regular Transformer with the same ~27M parameters and the same pipeline comes within ~5pp of HRM without hyperparameter tuning.	5 pp gap (Transformer vs HRM)	REPORTED	arcprize.org · Section on Finding 1 (HRM vs Transformer, Figure 3)	"a regular transformer comes within ~5pp of the HRM model without any hyperparameter optimization."
R4-176	On 8 core CLRS algorithms a standard Transformer (re-tuned) averages 42.34% OOD vs 81.30% for the Relational Transformer that adds edge vectors.	42.34 % OOD average, standard Transformer (8 core algorithms)	REPORTED	arxiv.org · Appendix, Table 13 (standard transformer ablation)	"Average 42.34% 81.30%"
R4-185	Iso-depth scaling law for looped LMs: looping a block r times is worth about $r^{0.46}$ unique blocks in validation loss ($\phi = 1$ would be full equivalence, $\phi = 0$ no capacity gain)	0.46 recurrence-equivalence exponent ϕ	REPORTED	arxiv.org · Abstract	"we fit a joint scaling law $L = E + A(\text{Nonce} + r\phi \text{Nrec})^{-\alpha} + B D^{-\beta}$ and measure a recurrence-equivalence exponent $\phi = 0.46$."

ROW	CLAIM	VALUE	STATUS	SOURCE	VERBATIM QUOTE
R5-017	NSA (trainable blockwise top-k sparse attention, Triton) reaches up to 9.0x forward and 6.0x backward speedup over FlashAttention-2 at 64k context	9.0 x forward speedup	REPORTED	arxiv.org · Section 6.1 Training speed, Figure 6	"our NSA achieves progressively greater speedups as context length increases, up to 9.0× forward and 6.0× backward speedup at 64k context-length."
R5-041	MoBA reaches up to 6.5x speedup prefilling 1M tokens vs FlashAttention full attention	6.5 x speedup	REPORTED	arxiv.org · Section 3.4 Efficiency, Figure 2	"In particular, it achieves a speedup ratio of up to 6.5x when prefilling 1M tokens."
R5-042	Quest (query-aware top-k KV page selection): up to 7.03x self-attention speedup, 2.23x end-to-end decode latency vs FlashInfer	7.03 x attention speedup	REPORTED	arxiv.org · Abstract	"We show that Quest can achieve up to 7.03× self-attention speedup, which reduces inference latency by 2.23× while performing well on tasks with long dependencies with negligible accuracy loss."
R5-049	torch.utils.flop_counter: operators without a registered formula (and no decomposition) add zero FLOPs		CODE	github.com · torch/utils/flop_counter.py module docstring, 'Counting semantics' (commit 958e982, 2026-09-28)	"Counts are produced by formulas in ``flop_registry``. Operators without a formula may be decomposed into registered operators; otherwise they add zero FLOPs."
R5-086	GCN on GPU: aggregation is memory-bound (L2 hit 6.87%, 2.35 DRAM bytes/op) while combination is compute-bound (L2 hit 82.5%, 0.01 DRAM bytes/op, 90% unit utilisation)	2.35 DRAM bytes per op (aggregation)	REPORTED	arxiv.org · Section 4 / Table 3 (Reddit dataset)	"The irregularity also leads to low L2 Cache Hit Rate (6.87%) and high DRAM Byte per Operation (2.35). In contrary, the Combination phase achieves 90% Computation Unit Utilization and 2.49 Executed IPC."
R5-092	At r=4 a 410M looped model matches a 580M non-looped model in loss but costs the training compute of a 1B non-looped model	410 M params	REPORTED	arxiv.org · Abstract	"For example, at r=4 a 410M looped model performs on par with a 580M non-looped model, but incurs the training cost of a 1B non-looped one."
R5-131	Random top-k edges leave essentially no empty 128x128 tiles: at N=512, k=8 the probability a given tile is empty is ~1e-128, so block-sparse kernels degrade to dense for unstructured learned graphs	1e-128 probability	DERIVED	arxiv.org · Inputs: R5-002/R5-003 (block-skipping semantics, BS=128); independence assumption ours	"The BlockMask unlocks the opportunity to save the work for fully-masked score matrix blocks without loading a large elementwise attention mask."
R6-001	Softmax is not Enough: any learned softmax circuitry must disperse as the number of items grows at test time, even for finding the maximum key; proven theoretically		THEOREM	arxiv.org · Abstract	"even for tasks as simple as finding the maximum key, any learned circuitry must disperse as the number of items grows at test time"
R6-006	ReSSFormer combines a recurrent reasoning unit with bounded depth and an adaptive sparse attention module		DESIGN	arxiv.org · Abstract	"Recurrent Reasoning & Memory Unit (R2MU) for iterative reasoning with bounded depth, Adaptive Sparse Attention Module (ASAM) for efficient and focused context selection"
R6-007	ReSSFormer's sparse attention selects, for each query, only the k highest-scoring keys (hard top-k routing)		DESIGN	arxiv.org · Section on ASAM (Adaptive Sparse Attention Module)	"For each query q_i q_i , only the k k highest-scoring keys are selected for attention computation"
R6-008	Explicit Sparse Transformer: explicit top-k selection of the most relevant segments in self-attention (2019)		DESIGN	arxiv.org · Abstract	"improve the concentration of attention on the global context through an explicit selection of the most relevant segments"
R6-010	Discrete NAR enforces hard attention in its recurrent processor and reports it as important for size generalisation, overcoming attention-weight annealing on arbitrarily large graphs		AUTHOR-CLAIM	arxiv.org · Section on the processor (hard attention paragraph)	"we enforce attention to be hard attention. We found this property important not only for interpretability but also for size generalization, as hard attention allows us to overcome the annealing of the attention weights for arbitrarily large graphs"
R6-011	Discrete NAR trains on SALSA-CLRS graphs of at most 16 nodes and tests on sparse graphs of 16 to 1600 nodes	1600 nodes (test max)	DESIGN	arxiv.org · Experimental setup	"The test set consists of sparse graphs of sizes from 16 to 1600 nodes"
R6-012	Neural Data Router (weight-tied Transformer with copy gate and geometric attention) reaches 100% length generalisation on compositional table lookup	100 % length-generalisation accuracy	REPORTED	arxiv.org · Abstract	"achieves 100% length generalization accuracy on the classic compositional table lookup task"
R6-016	ASentmax (alpha-entmax with adaptive scalable temperature) outperforms softmax, scalable softmax and fixed-temperature alpha-entmax, reaching up to 1000x length extrapolation	1000 x length extrapolation	REPORTED	arxiv.org · Abstract	"substantially outperforms softmax, scalable softmax, and fixed-temperature α -entmax baselines, achieving up to 1000× length extrapolation on synthetic benchmarks"

ROW	CLAIM	VALUE	STATUS	SOURCE	VERBATIM QUOTE
R6-017	Neural execution of graph algorithms: soft attention restricted to the input graph (GAT*) reaches 99.97% last-step reachability at 100 nodes (trained at 20), vs 91.51% for attention over the complete graph (GAT-full)	99.97 % reachability, last step, 100 nodes	REPORTED	arxiv.org · Table 1 (reachability), rows GAT* and GAT-full*	"GAT* (Veličković et al., 2018) GAT-full* (Vaswani et al., 2017) 93.28% / 99.86% 78.40% / 77.86% 93.97% / 100.0% 85.76% / 91.83% 92.34% / 99.97% 88.98% / 91.51%"
R6-018	LT2 Table 5, Transformer (dense attention): NIAH-Single-1/2/3 at 4096 tokens after pre-training at 2048 = 0.0 / 0.0 / 0.0	0.0 % NIAH-Single-1 at 4096 tokens	REPORTED	arxiv.org · Table 5 (1.3B, long-context evaluation), row Transformer	"Transformer 48.9 46.6 58.4 67.5 31.7 26.4 100.0 100.0 0.0 92.2 100.0 0.0 98.6 99.4 0.0"
R6-019	LT2 Table 5, GDN (gated linear recurrence (no routing)): NIAH-Single-1/2/3 at 4096 tokens after pre-training at 2048 = 99.8 / 49.8 / 34.2	99.8 % NIAH-Single-1 at 4096 tokens	REPORTED	arxiv.org · Table 5 (1.3B, long-context evaluation), row GDN	"GDN 32.7 40.0 28.3 63.5 25.7 24.5 100.0 100.0 99.8 100.0 93.8 49.8 83.8 68.4 34.2"
R6-020	LT2 Table 5, Mamba-2 (state-space recurrence (no routing)): NIAH-Single-1/2/3 at 4096 tokens after pre-training at 2048 = 62.0 / 11.8 / 13.4	62.0 % NIAH-Single-1 at 4096 tokens	REPORTED	arxiv.org · Table 5 (1.3B, long-context evaluation), row Mamba-2	"Mamba-2 30.7 39.1 23.7 64.3 25.1 28.5 100.0 99.6 62.0 100.0 53.8 11.8 95.8 87.4 13.4"
R6-021	LT2 Table 5, Looped Transformer (dense attention): NIAH-Single-1/2/3 at 4096 tokens after pre-training at 2048 = 0.0 / 0.0 / 0.0	0.0 % NIAH-Single-1 at 4096 tokens	REPORTED	arxiv.org · Table 5 (1.3B, long-context evaluation), row Looped Transformer	"Looped Transformer 52.8 49.4 61.7 68.2 33.6 28.1 100.0 100.0 0.0 94.6 100.0 0.0 99.2 99.8 0.0"
R6-022	LT2 Table 5, Looped GDN (gated linear recurrence (no routing)): NIAH-Single-1/2/3 at 4096 tokens after pre-training at 2048 = 99.8 / 53.2 / 35.8	99.8 % NIAH-Single-1 at 4096 tokens	REPORTED	arxiv.org · Table 5 (1.3B, long-context evaluation), row Looped GDN	"Looped GDN 34.9 41.8 30.6 64.7 27.0 25.9 100.0 100.0 99.8 100.0 96.4 53.2 85.6 71.0 35.8"
R6-023	LT2 Table 5, Looped Mamba-2 (state-space recurrence (no routing)): NIAH-Single-1/2/3 at 4096 tokens after pre-training at 2048 = 65.7 / 13.5 / 16.2	65.7 % NIAH-Single-1 at 4096 tokens	REPORTED	arxiv.org · Table 5 (1.3B, long-context evaluation), row Looped Mamba-2	"Looped Mamba-2 33.9 40.5 25.8 65.1 26.8 29.7 100.0 100.0 65.7 100.0 57.1 13.5 96.2 88.1 16.2"
R6-024	LT2 Table 5, Looped Hybrid (GDN+DSA) (linear recurrence + learned sparse top-k attention (routing)): NIAH-Single-1/2/3 at 4096 tokens after pre-training at 2048 = 91.4 / 77.6 / 60.3	91.4 % NIAH-Single-1 at 4096 tokens	REPORTED	arxiv.org · Table 5 (1.3B, long-context evaluation), row Looped Hybrid (GDN+DSA)	"Looped Hybrid (GDN+DSA) 51.6 48.0 60.4 66.9 33.0 28.4 100.0 100.0 91.4 100.0 100.0 77.6 100.0 99.6 60.3"
R6-025	LT2 Table 5, Looped Hybrid (Full+GDN) (full attention + linear recurrence (no routing)): NIAH-Single-1/2/3 at 4096 tokens after pre-training at 2048 = 93.5 / 81.0 / 63.7	93.5 % NIAH-Single-1 at 4096 tokens	REPORTED	arxiv.org · Table 5 (1.3B, long-context evaluation), row Looped Hybrid (Full+GDN)	"Looped Hybrid (Full+GDN) 53.1 48.9 62.0 67.8 34.0 30.2 100.0 100.0 93.5 100.0 100.0 81.0 99.8 99.8 63.7"
R6-026	LT2 authors: looped variants retain their base mixer's NIAH behaviour; recurrent backbones extrapolate to 4096 while dense-attention ones do not		AUTHOR-CLAIM	arxiv.org · Section on long-context evaluation (Table 5 discussion)	"they retain the qualitative NIAH behavior of their base versions – recurrent backbones extrapolate gracefully to 4096 while dense-attention ones do not"
R6-027	LT2 authors: looping helps subquadratic mixers more than full attention on the state-tracking + recall curriculum		AUTHOR-CLAIM	arxiv.org · Figure 9 discussion	"The striking pattern is that looping helps subquadratic mixers more than it helps full attention"
R6-028	LT2: Looped Full+GDN (no routing) also reaches stage 5 (n_max = 128), attributed by the authors to its linear-time GDN half		AUTHOR-CLAIM	arxiv.org · Figure 9 discussion	"Looped Full+GDN does too, but only because half its block is already linear-time GDN"
R6-032	LT2 curriculum: Looped GDN+Window solves only stage 3 at loop counts up to 4 and reaches stage 5 only at 8 loops; the n_max = 128 results are best-over-T		REPORTED	arxiv.org · Figure 9 discussion	"Looped GDN+Window is the most dramatic: stage 3 at T ≤ 4 , stage 5 at T = 8 ."
R6-034	Set-to-hypergraph model produces a new incidence matrix at every step from the previous step's edge and node vectors		DESIGN	arxiv.org · Section 4 (iterative refinement)	"we produce a new incidence matrix for step t based on the edge and node vectors from the previous step"
R6-035	Set-to-hypergraph model shares parameters between refinement steps, making it recurrent		DESIGN	arxiv.org · Section 4 (iterative refinement)	"By sharing the parameters between different refinement steps, we naturally obtain a recurrent model."
R6-036	Set-to-hypergraph model aggregates neighbouring edges weighted by the learned incidence probabilities, akin to message passing		DESIGN	arxiv.org · Section 4 (iterative refinement)	"This aggregation works akin to message passing in graph neural networks"
R6-037	LT2 setup prose: hybrids interleave the mixer with full attention in a fixed 4:1 ratio	4 mixer layers per full-attention layer (prose)	DESIGN	arxiv.org · Section 3 setup (Mamba-3 evaluation protocol paragraph)	"hybrids interleave that mixer with full attention in a fixed 4 : 1 ratio"
R6-038	LT2 Table 5 caption: hybrid models interleave with full attention in a 1:1 ratio, evaluated at 1.3B parameters	1 mixer layers per full-attention layer (Table 5 caption)	DESIGN	arxiv.org · Table 5 caption	"hybrid models interleave with full attention in a 1 : 1 ratio"
R6-039	LT2 Table 5 header labels the models 1.5B, while its caption says 1.3B	1.5 B parameters (Table 5 header)	DESIGN	arxiv.org · Table 5 header	"underline marks the second best. Model (1.5B)"

ROW	CLAIM	VALUE	STATUS	SOURCE	VERBATIM QUOTE
R7-004	LT2 synthetic state-tracking+recall task grows $n=m$ along a curriculum and reports the largest stage solved ($n_{\max}$), i.e. trained-at-size performance, not train-small/test-large extrapolation	256 tokens (max curriculum n)	DESIGN	arxiv.org · §3.6 Task and setup	"We tie $n = m$ and grow them together along the curriculum { 8 , 16 , 32 , 64 , 128 , 256 } ; a model advances once eval accuracy reaches 0.90 within a 100 k-step budget."
R7-005	On LT2 state-tracking+recall, the dense Looped Transformer plateaus at $n_{\max}=64$ at every loop count T	64 $n_{\max}$	REPORTED	arxiv.org · §3.6, Figure 9	"Looped Transformer and Looped Full+Window plateau at stage 4 ($n_{\max} = 64$) and never reach stage 5 at any T ."
R7-006	On LT2 state-tracking+recall, Looped NSA (learned top-k sparse attention in a loop) reaches $n_{\max}=128$, double the dense looped Transformer	128 $n_{\max}$	REPORTED	arxiv.org · §3.6, Figure 9	"three subquadratic variants — Looped NSA, Looped GDN+Window, and Looped GDN+NSA — all reach stage 5 ($n_{\max} = 128$)"
R7-007	LT2 compares sparse and dense looped arms at the same parameter budget		AUTHOR-CLAIM	arxiv.org · §3.6 Comparison across architectures	"a doubling of $n_{\max}$ over the global-attention baseline at the same parameter budget."
R7-012	LT2 attributes better NIAH extrapolation to the GDN+DSA loop despite it having no quadratic component		AUTHOR-CLAIM	arxiv.org · §3.7	"Looped Hybrid (GDN+DSA) tracks the Looped Transformer closely on the knowledge suite despite containing no quadratic component, and additionally extrapolates substantially better at NIAH-4096."
R7-032	Plain TRM fails to extrapolate on A5 state tracking: 45.8% at length 128 after training on up to 32 updates	45.8 % accuracy	REPORTED	arxiv.org · § length generalization, Figure 4	"plain TRM fails to extrapolate: at length 128, TRM and TRM with ACT enabled both obtain 45.8 % $\pm$ 3.9 % on A 5"
R7-033	Adding a causal 1D convolution to TRM raises A5 length-128 accuracy to 91.4%	91.4 % accuracy	REPORTED	arxiv.org · § length generalization, Figure 4	"Adding a causal 1D convolution layer substantially improves length generalization, reaching 91.4 % $\pm$ 2.3 % on A 5"