A Recurrent Reasoning Graft in a Frozen Language Model's Global Workspace Is Dominated by LoRA: A Negative Result

John Abrahams*

September 2026

Abstract

We tested whether multi-step reasoning can be installed into a frozen language model by grafting a small recurrent module with adaptive-computation halting into the model's measured global-workspace band (the "J-space") and training only the graft. The hypothesis had two parts: that a recurrent module placed at the workspace would recover reasoning the frozen base lacks, and that adaptive halting would let it think longer on harder problems and extrapolate beyond its training depth. Both parts fail. On the graft's own flagship task, a parameter-matched LoRA on the frozen base beats the graft on every held-out probe, by margins of 0.4 to 0.85 absolute accuracy, replicated across three seeds, and neither method extrapolates past the trained depth. A mechanism post-mortem explains the failure. The graft re-routes computation the frozen base already performs and never constructs a value absent from the prompt; its recurrence is dynamically inert on tasks it fails; and its halting head is structurally unable to detect convergence because a LayerNorm on the recurrent state deletes the one quantity that would carry the signal. Fixing the halting head makes it informative and buys no accuracy, because the extra rounds have nothing to compute. Swapping the recurrent core, adding working memory, varying recursion depth and schedule, and relocating the band do not change the outcome, and a TRM-faithful recursor trained from scratch on the same task loses to its own blocks run once by 0.7 to 1.0 absolute. We report the negative result together with four measurement findings that would affect anyone training a module inside a frozen transformer: grade the deployed EMA weights, not the raw ones; windowed checkpoint selection is blind to oscillating training; a per-step loss read after within-step optimizer updates measures memorization; and weight decay applied to normalization gains silently attenuates a graft's write into the frozen suffix. The broader lesson is procedural. The parameter-matched adapter baseline was the load-bearing control, an external reviewer predicted it would fail, and it was run last.

Introduction

Recurrent reasoning models such as the Tiny Recursion Model (TRM) (Jolicoeur-Martineau 2025) and the Hierarchical Reasoning Model (HRM) (Wang et al. 2025) obtain strong results on structured reasoning tasks from small modules that iterate and halt adaptively. Separately, interpretability work has identified a global workspace in language models: a low-dimensional band of the residual stream through which flexibly composed intermediate representations are broadcast (Anthropic 2026). Ablating the workspace contents collapses multi-step reasoning while leaving fluent generation intact.

*Independent researcher. Work conducted while a Master's student in Data Science at the University of Pennsylvania and an AI Labs intern at Ninth Wave. Correspondence: johnnycabrahams@gmail.com. Code, claims ledger, and per-claim evidence files available upon request.

These two results suggest an engineering hypothesis. If a frozen language model routes its flexible computation through a measurable band, then a small recurrent module inserted at that band and trained alone might recover reasoning without retraining the base, and adaptive halting might let it allocate more computation to harder inputs. The appeal is cost: the graft is about five percent of the base model's parameters and trains in hours on one GPU.

We built this system and tested the hypothesis over roughly two months and 48 recorded claims. An earlier draft of this work reported positive results: eight-hop in-context composition at 54 percent on a frozen Pythia-1B where the base scores near zero, transfer without loss to unseen entities, and depth extrapolation to twelve hops. Those numbers are real and reproduce. They are also uninformative about the hypothesis, because the comparison that decides it, a plain adapter of the same trainable size on the same frozen base, had never been run. When it was run, the adapter won on every probe by a wide margin.

This paper reports that result and the mechanism behind it. Our contributions are:

1. A controlled negative result. A parameter-matched LoRA (Hu et al. 2021) dominates the workspace graft on every held-out generalization probe across three seeds, and the graft's hypothesized differentiator, depth extrapolation through recurrence and halting, does not exist for either method (Section 4).
2. A mechanism post-mortem with independent evidence for each step: the graft routes rather than constructs; its recurrence is inert where it fails; its halting head is blind for a geometric reason we prove and then falsify-test; and no architectural variant we tried changes the outcome (Section 5).
3. Four measurement findings about training modules inside frozen transformers that survive the negative result and would have misled us had they not been caught (Section 6).
4. A procedural finding about how the program was run, which we think is the most transferable part (Section 8).

Related work

Global workspace in language models. The J-space construct and the Jacobian lens we use to locate it are due to (Anthropic 2026). That work's central ablation motivated grafting into the band rather than elsewhere. Our contribution was the graft and a band-locator built on the lens; we found that the locator's consensus rule fails on most models above 1B, where the band signal is diffuse, and that band placement never predicted graft quality.

Recurrent and latent reasoning. TRM (Jolicoeur-Martineau 2025) and HRM (Wang et al. 2025) train small recursive modules from scratch with deep supervision and adaptive halting. Universal Transformers (Dehghani et al. 2018) and ACT (Graves 2016) and PonderNet (Banino et al. 2021) are the older forms. Recurrent-depth scaling on a 3.5B model (Geiping et al. 2025) is the nearest architectural cousin at scale: the loop is present from the first pretraining step, and test-time accuracy on reasoning benchmarks rises with the number of recurrences. Looped transformers (Saunshi et al. 2025) make the depth claim precise: on synthetic reasoning tasks including p-hop induction, a k-layer block looped L times nearly matches a kL-layer model, with theory saying why. Coconut (Hao et al. 2024) is the latent chain-of-thought baseline, and a survey of the area is (Zhu et al. 2025). Our result does not bear on any of these in their own settings, and the reader should be clear about why. Those results place the loop in the model before pretraining, compare the looped model against a deeper non-looped one at matched parameters, and use tasks that need iterated

computation. We bolted a recursor onto a model that was pretrained without one, compared it against an adapter of the same size on the same frozen base, and used a task that the mechanism post-mortem shows is routing. The claim this paper kills is that a from-scratch-style recursor can be grafted into a frozen model and outperform an adapter. It says nothing about recurrent depth as a pretraining design, and Section 7 says what would.

Modules on frozen models. Universal Reasoner (Kim et al. 2025) trains a separate reasoning module and adds its logits to a frozen backbone's; it attaches in output space rather than mid-stack. Training-Free Looped Transformers (Chen et al. 2026) loop a mid-stack block of a frozen checkpoint with no training and report gains of one to three points on multiple-choice benchmarks. We did not run the training-free loop as a control; we note in Section 7 that we should have.

Parameter-efficient adaptation. LoRA (Hu et al. 2021) is the baseline that decides this paper. We note that the correct rank for a parameter match must be computed from the adapter library's actual parameter count per rank; the naive halving is wrong by a factor of two.

Method

Locating the band

We run the reference Jacobian-lens implementation of (Anthropic 2026) and compute its per-layer workspace signals: top-1 readout autocorrelation across positions, excess kurtosis of the readout logits, and effective dimension of the readout map. The band is the run of layers where the signals cross a baseline-plus-margin threshold. For Pythia-1B the band is layers 10 to 11, written $[10, 2)$. For larger models the autocorrelation signal alone is reliable; effective dimension ramps monotonically toward the output by construction and drags a consensus locator late. Bands are model-specific.

The graft

We excise the band and insert a weight-tied recurrent module operating on a latent answer register y and a scratchpad z over the frozen residual x:

$$z \leftarrow \mathrm{net}(x, y, z)\ (\times n), \qquad y \leftarrow \mathrm{net}(y, z).$$

The frozen prefix produces x; the frozen suffix decodes the refined y through the model's own unembedding. A halting head $q(y)$ predicts whether the current answer is correct. Both y and z pass through a LayerNorm on every update; this detail turns out to matter (Section 5.3). The recursor is warm-started by carving the excised band's own weights into it. On Pythia-1B the graft has 50,368,513 trainable parameters.

Training

Deep supervision with a one-step gradient, detached carry across rounds, a per-round optimizer step, an exponential moving average (EMA) of the graft weights, and ACT halting when $q > 0$. The EMA weights are the deployable artifact and the only ones we grade. Training uses masked targets: the answer positions are blanked in the input and appear only in the loss, so the graft must compute each hop rather than copy it. Teacher-forced targets produce a graft that scores near perfectly and never iterates (Section 6.1).

Tasks and probes

The flagship task is in-context entity multi-hop composition: facts of the form "A's R is B" are given in the prompt, shuffled with distractors, and the model must compose k relation hops. Training samples chains from a 19,525-entity pool over six relations with $k \leq 12$ and distractor counts in $\{3, 10, 20\}$. Evaluation is a held-out suite of four probes, 256 examples per cell, scored on the whole chain (every intermediate must be correct) on the EMA weights:

Probe	Meaning	Bar
A	in-pool entities, $k = 8$, three distractors	≥ 0.50
B	unseen entities (disjoint 400-entity holdout), $k = 8$	≥ 0.50, retention $\geq 100\%$
C	depth, $k = 12$ in-training and $k = 13, 14$ extrapolation	> 0.10 at 12
D	$k = 8$ under 40 distractors	≥ 0.15

Decode format. Every task in this paper is a masked-span fill-in, not autoregressive generation. The answer chain occupies a fixed span at the end of the prompt; those positions are blanked in the input, the model produces logits at every position in one forward pass (one pass per recursion round for the graft), and the span is read by causal-shifted argmax. An example is correct only if every token in the span is correct. No gold intermediate is ever fed back, so each answer position must resolve every hop before it within a single forward pass. This is the same non-autoregressive, whole-output-per-step format that TRM and HRM use natively, which is what makes a from-scratch TRM on this task a format-faithful comparison (Section 5.4). It also sharpens the LoRA result: 0.98 whole-chain accuracy at $k = 12$ means the adapted frozen stack composes twelve hops in one forward pass through sixteen layers, with no recurrence at all.

Two code tasks on Qwen2.5-Coder-7B distinguish routing from construction. In `hop`, a call chain must be followed through a program and the answer identifier is present verbatim in the prompt. In `trace_word`, the model must compute $(x \text{ op } c) \bmod 10$ and emit the answer as a word; the answer appears nowhere in the prompt.

The LoRA control

The control differs from the graft only in the trainable module. A PEFT LoRA of rank 96 on the frozen base's attention and MLP projections has 50,331,648 trainable parameters, within 0.07 percent of the graft. Data generator, entity split, training recipe, step count, masked targets, EMA, evaluation script, and scoring are shared. The graft additionally receives deep supervision over up to 16 rounds per step, which is more gradient signal per step than the LoRA's single forward pass.

The main result

Table 1 gives the full comparison on Pythia-1B, band $[10, 2)$, three seeds each, 10,000 steps.

Table 1: Graft versus parameter-matched LoRA on the held-out suite. EMA weights, whole-chain scoring, $n = 256$ per cell. Graft seed 0 is the ~7,000-step checkpoint from the original run; the graft plateaued at ~4,000 steps and seeds 1 and 2 at the full 10,000 steps confirm the plateau.

Probe	Graft s0	Graft s1	Graft s2	LoRA s0	LoRA s1	LoRA s2
A, $k = 8$	0.516	0.590	0.531	0.996	0.992	0.996
B, unseen $k = 8$	0.598	0.656	0.602	0.988	0.988	0.992
C, $k = 12$	0.211	0.199	0.184	0.984	0.984	0.992
C, $k = 13$ / 14	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.0 / 0.0	0.004 / 0.0	0.0 / 0.004
D, 40 distractors	0.215	0.141	0.144	0.988	0.992	0.988

Three observations.

The LoRA dominates on every probe. The gap is 0.4 to 0.85 absolute and exceeds the graft's seed spread (about 0.06 on probe A) by more than six times. Probe B rules out memorization: the LoRA holds at 0.99 on a disjoint entity pool.

The graft's hypothesized differentiator does not exist. Depth extrapolation beyond the $k = 12$ training limit is zero for both methods on every seed. The graft's ACT loop does spend more rounds on deeper chains (10 to 15 rounds at $k = 12$ against one round at $k \leq 4$), and this converts to nothing.

The graft is not broken. Its numbers reproduce the earlier positive draft (probe A around 0.55) to within seed noise. The positive draft was correct about the graft and silent about the baseline.

Why: a mechanism post-mortem

Each subsection reports an independent measurement. Together they explain Table 1 and rule out the rescues we could think of.

The graft routes; it does not construct

On the code tasks, the same model, band, and configuration succeed on `hop` and fail on `trace_word`. On `hop`, the graft beats the frozen Coder-7B by up to 30 times at depth 3 and extrapolates two depths past its training range (0.73 at depth 5 and 0.44 at depth 6 against a base of 0.016 and 0.000), on held-out identifiers, from a checkpoint paused at step 300. On `trace_word`, 1,100 steps at full recursion asymptote exactly at chance.

Linear probes explain the split. After training on `trace_word`, the answer is not decodable anywhere in the graft's residual stream: four probes across two checkpoints and two round counts read 0.125 to 0.138 against chance 0.10 and a Bonferroni gate of 0.151, flat across all 15 capture points. An operand control, the same probe on the same activations recovering an input token, reads 1.000 in every run. The readout works; the value is absent. On the frozen base alone, across all 30 residual sites, the answer is likewise never decodable (0.08 to 0.13 at every site, operand control 1.000 everywhere), so there is no better band to move to.

A grokking-regime test closes the memorization route. On a fixed pool of 256 `trace_word` examples with weight decay 1.0, the graft memorizes the pool by step 1,500 and is run to 32 times that point. Zero of 185 held-out evaluations approach the bar; the graft learns the literal prompts and never the ten-entry fact table underneath them.

The dissociation is clean: where the answer token is in the prompt, the graft learns to route attention to it and does so well; where the answer must be constructed, the graft never learns it at any budget, schedule, or band. Entity multi-hop, the flagship task, is a routing task. A LoRA on the base's own attention does routing better than a frozen-band recursor at equal cost.

Recursion computes on one substrate and is inert on another

Forced-round curves measure accuracy at every round with halting ignored, avoiding the degeneracy of comparing halted accuracy to a one-round readout. On Pythia-2.8B, entity task, the recursor genuinely computes across rounds: at depths 11, 12, and 14 accuracy is exactly zero after one round and reaches 0.98, 0.91, and 0.78 by round four to six. This is the strongest evidence in the program that the module iterates.

On Coder-7B the picture inverts. On `trace_word` the recursion reaches an exact fixed point after one round: cosine between successive states is 1.0000 from round 2 for both y and z, and accuracy is byte-identical across all eight rounds. The halting head on the same checkpoint had requested maximum compute for 800 consecutive training steps. On `hop`, where the graft works, more rounds hurt: round 2 is optimal and depth-6 accuracy falls from 0.44 to 0.27 by round 8.

Recursion is therefore neither necessary nor sufficient. Where it helps (a weak base on a routing task), a LoRA helps more. Where the task needs construction, the recursion does not run.

The halting head is structurally blind

The halting head was miscalibrated from the first week. A hand-picked constant threshold beats the trained head in four of four cells, and a frozen-base surprise signal does no better. We treated this as tuning. It is geometry.

The recurrent state passes through a LayerNorm on every update, so $\|y_t\|$ is pinned to $\sqrt{d}$ at every round (measured 45.25 on Pythia-1B, 22.627 on a 70M model, identical at every round and cell). A state on a sphere cannot grow or shrink, only rotate, so "how much did I move" is purely an angle between two consecutive states. The halting head reads one state, $q(y_t) = w^\top y_t + b$, never y_{t-1}. It cannot compute the angle. Measured, 98.8 percent of the pre-norm size variation is destroyed, and $p_{\text{halt}} = 0.992 \pm 0.003$ whether accuracy is 0.49 or 0.15.

The same LayerNorm also rules out a halting mechanism we had wanted to test: consolidation-as-integration in the sense of Bhasin, Raymond and Goldman (Bhasin et al. 2024), where integration stops when a fast site returns to baseline. The fast state cannot decay, and the learned baseline term turns out to receive no gradient at all under the no-grad warm-up schedule, so it is a permanent zero vector. Fast-state settling, measured angularly, exists but lags correctness by 2.3 to 3.1 rounds in every cell on Pythia-2.8B and is an example-independent clock on Coder-7B.

The account makes a prediction: feed the head the step, and it becomes informative. We ran it. With q over $(y_t, \cos\theta_t, \|y_t - y_{t-1}\|)$, the standard deviation of p_{halt} rises 8 to 30 times, p_{halt} falls monotonically with depth, and its correlation with correctness is positive at every depth with nonzero accuracy. Rounds used rise from 1 to 16 with depth. Accuracy is unchanged: probe A

0.51, C 0.14, D 0.16, inside the carved range. An informed halting head spends the rounds and converts them to nothing, because there is no computation for the extra rounds to perform.

Nothing in the design space rescues it

Every variant we ran lands inside the graft's seed range or below it.

- **Recursion configuration.** Four configurations on `trace_word`, including TRM's exact published schedule (6 low cycles, 3 high cycles, 16 halt steps), are four nulls.
- **Recurrent core.** Replacing the carved transformer layer with a parameter-matched Mamba-2 hybrid core (Wang and Reid 2026) on the entity task gives A 0.49, B 0.57, C 0.12, D 0.17 at 5,000 steps; the in-training curve tracks the carved runs step for step.
- **Inference-time width.** PTRM-style noise rollouts with the halting head as selector (*PTRM* 2026) are a structural null on `trace_word` (the state annihilates the perturbation) and the bottleneck on `hop` is the selector, which a constant beats.
- **Placement.** An off-band graft at layer 2 raises probe A to 0.66 while halving depth extrapolation and losing distractor robustness; the measured band is better for depth and worse for easy accuracy. On Coder-7B no layer holds the `trace_word` answer, so no relocation can help there. The random-band control that would test whether the measured band matters at all was never run.
- **TRM from scratch on the task itself.** Because the task format is TRM's native format (Section 3.4), we can ask directly whether a TRM-style recursor trained from scratch, with no frozen base and no graft, beats a plain transformer here. Our first attempt (a sandwich of four prefix blocks, a TRM core, two suffix blocks, about 6M parameters) used pre-LayerNorm blocks and the same external LayerNorm on the recurrent state as the graft. The recursive core scored 0.28, 0.17, 0.0, 0.0 at $k = 1$ to 4 and a one-pass control with the same blocks scored 0.50, 0.48, 0.27, 0.09. We then rebuilt the arm to match canonical TRM: post-norm RMSNorm and SwiGLU blocks, no external state norm, truncated-normal state buffers, the published 6/3/16 cycle schedule, halt exploration, and grading at full depth as well as with halting. Two things happened. The one-pass control with the faithful blocks reached 1.00, 1.00, 0.99, 0.99, 0.97, 0.77 at $k = 1$ to 6, so the earlier block design had been costing it most of its accuracy. The faithful recursor, at TRM's own learning rate and warmup, reached 0.31, 0.20, 0.02, 0, 0, 0; with four supervision rounds instead of sixteen it reached 0.30, 0.30, 0.01, 0, 0, 0; and at the control's learning rate with sixteen rounds it collapsed to the unigram floor at step 2000 and stayed there. Full-depth and halted accuracy were identical on every recursive arm. Nothing extrapolated past the trained depth on any arm. On this task, the same blocks run once beat the same blocks run recursively by 0.7 to 1.0 absolute.
- **Scale.** Pythia-2.8B at a relocated band reaches 1B parity, and a wider recursor there clears all four probes (A 0.92, B 0.94, C 0.89, D 0.97), but these runs predate the LoRA control and were never compared against it. Given the 1B result and the routing mechanism, we do not expect the ordering to change.

What survives

Adaptive computation is a property of the target

Under teacher forcing (gold chain in the input), the graft reaches accuracy near 1.0 at every depth while using exactly one round everywhere: it copies. Under masked targets on the identical architecture, accuracy falls with depth, rounds rise with depth, and halted accuracy exceeds the one-round

readout at depth 4 and beyond. The dissociation held through every tightening of the evaluation. Whether a recurrent module "thinks" or copies is decided by whether the target is copyable, not by the module. We consider this the one capability-side finding worth carrying forward, with the caveat from Section 4 that the thinking it induces does not beat an adapter.

Four measurement findings

Each of these produced wrong numbers before it was caught, and each applies to anyone training a module inside a frozen transformer.

Grade the deployed EMA weights. Training kept an EMA of the graft because the EMA is what ships, but the checkpoint on disk saved raw weights, and the evaluation suite had graded raw weights for weeks. On one recipe, raw weights scored A 0.42, C 0.07, D 0.0 and EMA weights on the same run scored 0.54, 0.22, 0.36. A working method was classified as a failure by a checkpoint-writing bug.

Windowed checkpoint selection is blind to oscillation. On `trace_word`, training loss oscillates with a period of 10 to 20 steps. Best-checkpoint selection averaged loss over a 10-step window, so every window straddled a trough and a peak and the minimum windowed loss (0.36 to 0.51) never captured the instantaneous troughs (0.001 to 0.03). Every grade in the program's history up to that point had evaluated the wrong checkpoint. (The troughs, once graded, sat at chance on held-out probes; this fix changed which checkpoint was wrong, not the verdict.)

A loss read after within-step updates measures memorization. Deep supervision reused each batch across rounds with an optimizer step inside the loop, and the logged loss was read from the last round, after up to seven updates on those exact examples. On an untrained graft the metric read 0.79 against a chance of 2.30. The "oscillation" above was partly this: each step fit its own batch. Read the loss on the first round or on a fresh batch.

Weight decay on normalization gains attenuates the write. A single AdamW parameter group applied weight decay to LayerNorm gains, biases, and initial states. At the default 0.01 this is harmless. At 1.0, introduced for the grokking regime, the output LayerNorm gain decayed as $\exp(-2 \cdot \text{step} \cdot \text{lr} \cdot \text{wd})$ to three decimals, progressively silencing everything the graft wrote into the frozen suffix. The graft sat at chance for 3,000 steps and looked incapable of memorizing 256 examples; with 1-D parameters excluded from decay and nothing else changed, it memorized them by step 1,300. Every high-decay result before the fix was void.

Limitations of the negative

The LoRA comparison is on one base family (Pythia-1B), one band, one synthetic task family. The Coder-7B and Pythia-2.8B grafts, which score much higher in absolute terms, were never run against a matched LoRA. We think the routing mechanism makes the outcome predictable, but it is inferred there, not measured.

The from-scratch comparison is one seed per arm, 15,000 steps, about 6M parameters against TRM's much larger compute, with learned absolute positions where TRM uses rotary ones, a halting head that reads the last answer slot, and softmax rather than stablemax cross-entropy. Those deviations remain. The five architectural confounds of the first from-scratch attempt (external state LayerNorm, pre-norm blocks, zero-initialized states, one-logit halting without exploration, a shortened schedule) were removed in the faithful rebuild, which widened the gap rather than closing

it. Attention is causal in every arm. A bidirectional pair was written and, after the causal pair's 0.7 gap, deliberately not run: hop i depends only on hops before it, the adapted frozen model composes twelve hops causally, and no plausible masking effect closes a gap of that size. The recursive arms' failure mode at TRM's learning rate deserves a note: their within-batch training loss reached zero while held-out accuracy stayed near 0.3, which is per-batch memorization from sixteen optimizer steps on the same 64 examples, the deep-supervision recipe applied at a data scale far below TRM's. Whether the recursor would win at TRM's data scale is not tested here.

What this negative does not say about recurrent depth. The recurrent-depth and looped-transformer results (Geiping et al. 2025; Saunshi et al. 2025) differ from our setting on three axes at once: the loop is trained in from the start rather than grafted onto a frozen base; the comparison is against a deeper non-looped model at matched parameters rather than against an adapter on the same base; and the tasks require iterated computation rather than routing. Any one of the three could account for the opposite outcomes, so nothing here contradicts them and nothing here supports them. The from-scratch comparison is the closest we came, and it is one seed at about 6M parameters and 15,000 steps, where the looped arm lost to its own blocks run once; (Saunshi et al. 2025) report the looped arm matching a deeper model on p-hop induction at a data scale we did not reach, so the tension is a scale question we leave open. The experiment that would bear on recurrent depth directly is to take a pretrained recurrent-depth checkpoint (Geiping et al. 2025), run our probes on it, and confine a matched adapter either to the looped block or to the non-looped prelude and coda. If the loop is where the model's iterated computation lives, adapters inside it should behave differently from adapters outside it on the depth probes. We did not run that experiment; it is the successor project's first architectural test.

Two controls the design called for were never run: a graft at a random band, and the training-free looped block of (Chen et al. 2026). The first would test whether the measured band matters at all; the second would test whether the trained recursor beats zero training. We expect both to make the graft look worse, not better, but that is an expectation.

An earlier draft reported a recursor-depth sweep (2, 4, 8 applications per round) moving no probe. Its evidence file could not be located during the audit and we do not rely on it.

Finally, the LayerNorm on the recurrent state is a confound for the entire recurrence story. It caused the halting blindness, forbids the consolidation dynamics, and differs from TRM's internal post-norm design. A graft without it might iterate differently. We did not build one, because Section 5.1 says the task would still be routing and the LoRA would still win.

Discussion: how the program was run

The hypothesis was falsified by a control that costs about ten dollars and that was the first thing a skeptic would ask for. It was run in the ninth week. Three things went wrong in the process, and we think they generalize.

The baseline was a citation. LoRA appeared in the related-work list from the first draft as a contrast ("unlike LoRA, we replace a band and freeze the base") and never as an arm. Framing a method as different from the baseline is not the same as beating it.

Positive numbers absorbed attention. The graft did learn something, and learning something produces a stream of real, publishable-looking results: generalization to unseen entities, depth extrapolation to twelve hops, iteration curves, a placement effect. Each was true. Each was also

uncontrolled, and the program spent its effort hardening them (seeding, EMA grading, checkpoint selection) rather than asking whether they were the right numbers. The earlier draft of this paper had a six-item contributions list and no baseline.

The prediction was on the record. An external review on 2026-07-17 called the novelty thin and predicted that both the measured-versus-random-band gate and the graft-versus-LoRA gate would fail. The LoRA gate was run three weeks later and failed as predicted. The program had written the gate into its own roadmap as the condition for continuing; it had not scheduled it.

The claims ledger that organized the work (48 claims, each with a status and an evidence file, refuted claims kept in place and struck through) is what made the negative clean once it arrived. It is also what let the positive results accumulate without a baseline for two months. A ledger records what was tested. It does not tell you what to test first.

Conclusion

A small recurrent module with adaptive halting, grafted into a frozen language model at its measured global-workspace band, learns to route the frozen model's existing computation and nothing more. A LoRA of the same size routes better. Neither extrapolates past its training depth. The halting head cannot see convergence because the state it reads has been normalized onto a sphere, and giving it sight changes nothing because there is nothing for the extra rounds to compute. Trained from scratch in canonical TRM form, the recursor still loses to the same blocks run once. What the program produced that is worth keeping is a clean dissociation between copying and computing as a property of the training target, the finding that an external LayerNorm on a recurrent state plus pre-norm blocks silently cost a small transformer most of its accuracy, four measurement bugs that will bite anyone doing this kind of surgery, and a reminder that the boring baseline goes first. The negative is about grafting a loop onto a model pretrained without one. Whether a loop pretrained in from the start does what its proponents claim is a different experiment, and we name it in Section 7.

References

Anthropic. 2026. *Verbalizable Representations Form a Global Workspace in Language Models*. Https://transformer-circuits.pub/2026/workspace/index.html.

Banino, Andrea, Jan Balaguer, and Charles Blundell. 2021. *PonderNet: Learning to Ponder*. https://arxiv.org/abs/2107.05407.

Bhasin, Brandon J., Jennifer L. Raymond, and Mark S. Goldman. 2024. "Synaptic Weight Dynamics Underlying Memory Consolidation: Implications for Learning Rules." *Proceedings of the National Academy of Sciences*.

Chen, Li, Liang, Lao, and Liu. 2026. *Training-Free Looped Transformers*. https://arxiv.org/abs/2605.23872.

Dehghani, Mostafa, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. *Universal Transformers*. https://arxiv.org/abs/1807.03819.

Geiping, Jonas et al. 2025. *Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach*. https://arxiv.org/abs/2502.05171.

Graves, Alex. 2016. *Adaptive Computation Time for Recurrent Neural Networks*. https://arxiv.org/abs/1603.08983.

Hao, Shibo et al. 2024. *Training Large Language Models to Reason in a Continuous Latent Space*. https://arxiv.org/abs/2412.06769.

Hu, Edward J., Yelong Shen, Phillip Wallis, et al. 2021. *LoRA: Low-Rank Adaptation of Large Language Models*. https://arxiv.org/abs/2106.09685.

Jolicoeur-Martineau, Alexia. 2025. *Less Is More: Recursive Reasoning with Tiny Networks*. https://arxiv.org/abs/2510.04871.

Kim, Jaemin, Hangeol Chang, Jongwoo Hwang, Beomsu Kim, and Jong Chul Ye. 2025. *Universal Reasoner: A Single, Composable Plug-and-Play Reasoner for Frozen LLMs*. https://arxiv.org/abs/2505.19075.

PTRM: A Probabilistic Extension of the Tiny Recursion Model. 2026. https://arxiv.org/abs/2605.19943.

Saunshi, Nikunj, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. 2025. *Reasoning with Latent Thoughts: On the Power of Looped Transformers*. https://arxiv.org/abs/2502.17416.

Wang, Guan, Jin Li, Yuhao Sun, et al. 2025. *Hierarchical Reasoning Model*. https://arxiv.org/abs/2506.21734.

Wang, and Reid. 2026. *Tiny Recursive Reasoning with Mamba-2 Attention*. https://arxiv.org/abs/2602.12078.

Zhu, Rui-Jie, Tianhao Peng, Tianhao Cheng, et al. 2025. *A Survey on Latent Reasoning*. https://arxiv.org/abs/2507.06203.

Appendix: claim index

Every quantitative statement above traces to a row in the project's claims ledger and an evidence file with raw JSON. The mapping, for readers with access to the repository:

Section	Ledger claims
4, main result	C43, C44
5.1, routes not constructs	C29, C30, C39, C40
5.2, recursion	C26, C36, C37
5.3, halting head	C35, C38, C42, C46, technical note 2026-08-12
5.4, design space	C4, C31, C41, C45, C47, C48, C49, C13, C16

Section	Ledger claims
6.1, target property	C2
6.2, measurement	C3, C23, C27, C34
7, limitations	C5, C14, C17, C49, ROADMAP history