AGREEMENT RATHER THAN NORMALISATION KEEPS DEPTH REUSABLE IN LOOPED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped language models apply one block of layers repeatedly, and the design pays off only if depth is reusable: loop steps beyond the trained budget must remain usable at inference. We ask whether that rests on normalising the state between loop steps, as the released Ouro models do, or on agreement between the loop a model is trained with and the loop it is run with. On released weights this loop-boundary normalisation can be removed without adding parameters and with the readout path bit-identical, and the operation is an identity at one loop step, which serves as an implementation check. Removing it takes Ouro-1.4B and Ouro-2.6B to zero accuracy by the trained budget on a balanced binary task (chance 0.500). State-norm growth, the failure that prior accounts predict, is not the cause: the norm stays within a factor of 2.021 while the head commits to a single wrong token. Yet in 19M-parameter models trained from scratch on two synthetic tasks, 16 seeds per configuration, training without the normalisation is never worse than training with it and extrapolates better: the two are closest at the trained depth and separate on both sides of it (Cliff's δ up to 0.867 beyond the budget). Both results follow if agreement, and not the normalisation itself, is what keeps depth reusable. The damage sits in the readout together with the last layer of the block: in the models trained from scratch, refitting the head over the frozen modified loop recovers a third of the lost accuracy and plateaus, while refitting the last block layer with it recovers all of it. Finally, comparing looped with stacked depth at an equal inference loop count is confounded by unequal overshoot of the trained budget; at matched overshoot the advantage of looping vanishes on one task and shrinks on the other.

1 INTRODUCTION

Looped language models apply one stack of layers to the state r times, with r chosen at inference and expected to remain useful beyond the number of loop steps r_0 used in training. Without that, looping only saves parameters at a fixed depth. Which part of the architecture keeps loop steps reusable past the trained budget, and which part ties the model to r_0, is usually left open.

Two released models happen to differ in just this detail. Both normalise inside their recurrent block, but Ouro (Zhu et al., 2025) also renormalises the state at the end of every loop step, with the module that normalises the readout input, while Huginn (Geiping et al., 2025) applies that module once per forward pass. We call this placement the loop-boundary normalisation and determine it by instrumenting the models (Sec. 3.2); it makes an architectural hypothesis testable without training. Restricting Ouro's model-level normalisation to the final loop step leaves the readout path bit-identical and changes only the state that re-enters the loop, adds no parameters, and is an identity at $r = 1$, which checks the implementation.

One rearrangement cannot separate two explanations. The first is that boundary normalisation is required, because without it the state diverges and the loop stops being reusable. The second is that agreement is required, because the readout is calibrated on the states that training produces, and any change to the loop topology moves its input off that distribution, in either direction. On released weights both predict the same outcome, since the training topology there is fixed, so we repeat the experiment on models trained from scratch with the topology set before training: fifty

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.

models per task, 16 seeds per configuration on the normalisation axis, with the training seed as the unit of analysis and rank tests over seeds.

The answer has two parts, and the two halves of the study supply one each. Removing boundary normalisation from a model trained with it breaks it within a few loop steps, so agreement is necessary. Training from scratch without the normalisation is never worse than training with it and is better at every depth other than the trained one, so given agreement the normalisation is a cost and not a benefit: the configuration trained with it peaks at its trained depth and degrades on both sides, although its training and inference loops agree. Neither part is what the literature predicts. The nearest prior account expects an architecture with a normalisation between loop steps to avoid the failure by construction, and the residual-growth account expects failure through the scale of the state. The state norm of the damaged model changes by at most a factor of 2.021 over the whole grid, while readout entropy rises from 0.786 to 12.470 bits on a one-bit task and then falls back as the head commits to a single token outside the answer alphabet.

Our contributions are the following.

1. An exact rearrangement of the loop-boundary normalisation on released weights (Huginn-3.5B, Ouro-1.4B, Ouro-2.6B), a measured topology audit, an identity check at $r = 1$, and measurements of readout entropy, logit margin, state norm per loop step and input-to-prediction mutual information (Sec. 4.1).
2. The causal version of the same question on models trained from scratch with the topology fixed before training, on two task families, 16 seeds per configuration, an exact rank test and Cliff's δ (Sec. 4.3).
3. A separation of "normalisation is required" from "the training and inference topologies must agree" in favour of the second, on four measurements: the state scale barely moves while accuracy collapses, the readout leaves the answer alphabet and then collapses onto a constant, a readout refit over a frozen loop plateaus at a third of the loss while refitting the last block layer with it returns all of it, and training without a boundary normalisation is better although both configurations agree with their own loop (Sec. 4.2–4.4).
4. A methodological result on comparing depth in the loop against depth in the stack: comparing at equal absolute K favours whichever configuration has the larger trained loop count r_0, although it is the one with less inference compute; at equal budget overshoot K/r_0, which is also equal compute, the advantage of looping is not detectable on `circuit` and shrinks without closing on `hops` (Sec. 4.5).

2 Related work

Looped models and depth as an inference-time resource. Reusing one block instead of a stack goes back to the Universal Transformer (Dehghani et al., 2019), and Giannou et al. (2023) showed that a looped transformer can simulate arbitrary programs at a constant parameter count. Three recent systems raised this to language-model scale: Huginn-3.5B (Geiping et al., 2025), a prelude, core and coda topology trained on 800B tokens; Ouro (Zhu et al., 2025), which moves the loop into pretraining on 7.7T tokens with an entropy regulariser on the depth distribution and a learned exit gate; and retrofitted recurrence (McLeish et al., 2025), which converts an already pretrained model. Around these sit elastic-depth variants (Jeddi et al., 2026), training-free wrappers (Chen et al., 2026), memory-decoupled implementations (Conchello Vendrell et al., 2026), scaling studies (Huang et al., 2026; Gao et al., 2026) and analyses of looping as a substitute for chain of thought (Kohli et al., 2026; Fan et al., 2026; Zhang, 2026a).

Where the normalisation sits. Blayney et al. (2026) tabulate the block equations of the released models: Huginn normalises the residual stream after each sublayer while Ouro normalises the attention and MLP outputs, and both apply a normalisation after the loop. Normalisation inside a loop has since been analysed as an implicit gain control on a local Lipschitz constant (Buehlmaier, 2026), as the condition for reachable, input-dependent fixed points (Labovich, 2026), through pre-norm against post-norm placement (Movahedi et al., 2026), and through which iterates converge at all (Zhang, 2026b; Logan, 2026; Fein-Ashley & Rashidinejad, 2026). Past the trained loop count, Kuo et al. (2026) trace out-of-distribution variance to a spurious correlation between sequence length

and loop count, and halting gates and trajectory readouts are diagnosable in their own right (Popescu et al., 2026; Lam, 2026). None of this work holds the readout fixed and moves the loop-boundary normalisation on released weights.

The nearest neighbour. The closest work is on the readout blind spot (Sharma & Vu, 2026): dense per-iteration cross entropy supervises only the state variables visible through a scale-invariant readout, so a pre-norm residual recurrence keeps updating the radial scale unseen by the loss, and at 44M and 129M parameters the final state norm reaches thousands. Its intervention on released weights is a scalar rescaling of Ouro-1.4B's final hidden state before the readout, not a change of loop topology. That work varies the readout normalisation and holds the loop-boundary normalisation fixed; we do the reverse, so the two studies cover orthogonal axes of one design space. Its limitations section names our condition as open, expecting inter-loop normalisation to avoid the failure by construction; we measure that case, and training with a boundary normalisation makes extrapolation worse (Sec. 4.3). Both that account and the residual-growth account predict failure through state-norm growth, which our measurements do not support: the state scale barely moves while the readout leaves the answer alphabet and then settles on a constant.

Depth in the loop against depth in the stack. Schwethelm et al. (2026) fit a joint scaling law over an iso-depth pretraining sweep and report a recurrence-equivalence exponent of 0.46, so shared recurrences raise validation loss at matched training compute, and Oncescu et al. (2026) approach the same trade-off from effective depth and decoding cost. Both read off one quality number at fixed training compute or effective depth; we ask the inference-time question, comparing three placements of a matched depth against the loop count used at inference (Sec. 4.5).

Abrupt emergence of in-context lookup. Induction-head formation shows up as a bump in the training curve (Olsson et al., 2022), and associative recall with many candidate keys formalises the harder variant (Arora et al., 2023); we observe such a transition on single-hop lookup over an in-context map (App. A.4).

3 Setup

3.1 Models

The released checkpoints, Huginn-3.5B, Ouro-1.4B and Ouro-2.6B, carry the scale; the causal claims rest on models trained from scratch, from 19M parameters at six layers to 76M at twenty-four, where the topology is a flag set before training. The division is deliberate: sixteen independent training runs per configuration are not feasible at 3B parameters, and without them there is no rank test over seeds, so the two halves answer different questions (Table 1). We write r for the inference-time loop count of released models and K for that of models trained from scratch; r_0 is the trained budget in both cases.

3.2 The axis: loop-boundary normalisation

By *loop-boundary normalisation* we mean a normalisation applied to the state that leaves one loop iteration and enters the next. It is distinct from the normalisations inside the block, which both released models have and which Blayney et al. (2026) tabulate.

We determine the placement by instrumentation. Every normalisation module in a loaded model is instrumented and its calls are counted over one forward pass at two loop counts; a module whose count grows with the loop count acts inside the loop, and a module called the same number of times acts on the readout path. On released weights the audit returns `norm_in_loop` for Ouro, with the model-level normalisation that produces the readout input called once per loop step, and `readout_only` for Huginn. Since the instrument counts calls to normalisation *modules*, `readout_only` means that no such module is called per loop step, not that no normalisation acts there; this matters only for the Huginn direction (Sec. 6).

Two operations follow. For Ouro we restrict the model-level normalisation to the final loop step, so the readout path is bit-identical and only the state re-entering the loop changes; this is the operation

our results rest on. For Huginn we apply the readout-path normalisation to the state before it re-enters the block (Sec. 4.1 and Sec. 6). In the models trained from scratch the same axis is a flag set before training, with a dedicated normalisation applied between iterations and skipped on the final one; unlike the released case, that normalisation has its own parameters instead of sharing them with the readout. At one loop step every one of these operations is an identity by construction, and we use that identity check as a gate: an arm whose $r = 1$ numbers differ from its baseline is discarded.

3.3 TASKS

We use three families of synthetic tasks with an exact per-instance depth $d^\star$ and an independent width control w. The answer is a single token, so arg-max metrics are defined without decoding. The family `circuit` evaluates a Boolean circuit of depth $d^\star$; `hops` follows $d^\star$ pointers through an in-context map with 26 possible answers; `paragg` computes the parity of a subset of answers. Instances are generated on a grid of cells $(d^\star, w)$; a *stratum* is one cell and a *split* the evaluation instances of a family. The grid is materialised in two tiers, 100 instances per cell over the full grid and more on the pre-registered primary cell. The generator and the verifier are separate code, an instance that fails verification does not enter a split, and both binary families are balanced by construction at 50/50 in every stratum.

3.4 METRICS AND STATISTICS

Accuracy is the arg-max over the model's full vocabulary compared with the gold answer token, reported against an explicit floor equal to the larger of random guessing within the answer alphabet and the best constant answer: 0.500 for the binary families and, for `hops`, 0.038 on released models, which answer over the 26-letter alphabet, and $1/w$ on models trained from scratch on a single cell of width w; where the majority answer of a split beats chance, that majority floor is reported instead. The floor is defined within the alphabet and the arg-max is not, which matters for one reading: on a balanced binary family a constant answer within the alphabet scores 0.500, so an accuracy of zero means that the top token has left the answer alphabet, and the format statistics of App. A.7 keep that case separable from a wrong answer within it. We also report readout entropy in bits, the top-1 minus top-2 logit, the state norm per loop step, and, for collapse, the mutual information between input and arg-max prediction with the Miller-Madow correction, the modal share and the number of distinct predictions. On released weights we use an exact McNemar test on paired instances with a cluster bootstrap over (family, $d^\star$, data seed) and a Holm correction; on models trained from scratch the unit of analysis is the training seed, 16 per configuration on the normalisation axis and 6 on the depth axis, with an exact Mann-Whitney test and Cliff's δ; averaging over instances would treat one run as a thousand observations.

A competence gate is fixed in advance: a pair of arm and family enters the analysis only if peak accuracy reaches 0.30 in absolute terms and exceeds the floor by more than 0.05, and the pairs that fail it are reported as such.

4 RESULTS

4.1 THE REARRANGEMENT IS ASYMMETRIC

The identity check passes exactly: at $r = 1$ the difference in accuracy and in readout entropy between an arm and its rearranged counterpart is 0.0000 in all nine pairs of arm and family zero-shot and all four few-shot, so everything at $r > 1$ is a property of the loop and not of weight loading or answer scoring.

Removing the boundary normalisation from Ouro breaks the model within the first loop steps. On `circuit` with four in-context examples, Ouro-2.6B goes from 0.530 at $r = 1$ to 0.500 at $r = 2$, 0.058 at $r = 3$ and zero, to three decimals, from the trained budget onward, while the unmodified model peaks at 0.642 at $r = 4$ and still reads 0.583 at $r = 32$ (Fig. 1; full grid in Table 9). Ouro-1.4B breaks earlier, at 0.090 at $r = 2$ and zero at $r = 3$, so the damage is complete within two to three loop steps. On `hops` the same arm reaches 0.000 by $r = 8$ while the unmodified model peaks at 0.166 against the majority floor of 0.072 (chance over the alphabet is 0.038).

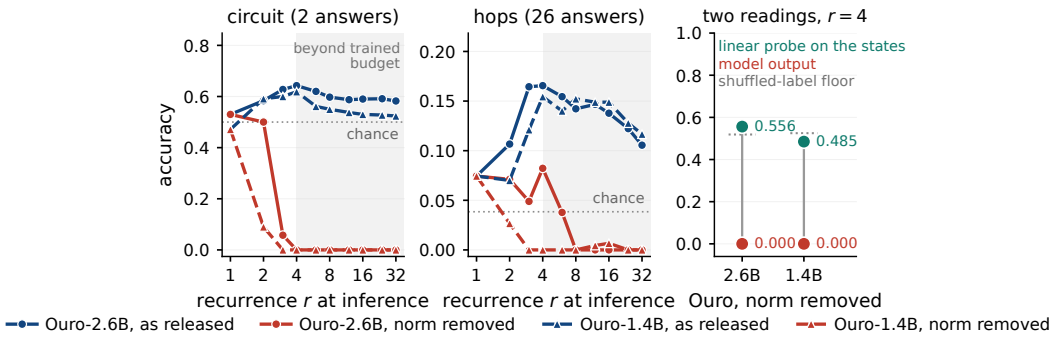

Figure 1: Rearranging the loop-boundary normalisation on released weights, four in-context examples in the prompt; blue as released, red with the boundary normalisation removed, line style marks the model. Left and middle: accuracy against the inference-time loop count r; the shaded band lies beyond the trained budget $r_0 = 4$ and the dotted line is chance inside the answer alphabet (on `hops` the majority floor, 0.072, is higher; the y ranges differ). Right: the same states of the arm with the normalisation removed, at the trained budget on `circuit`, read by the model's own head and by a linear probe fitted to them, with the floor from a probe fitted to shuffled labels. The probe sits at its floor while the output sits at zero, below anything an answer within the alphabet would give (Sec. 4.2).

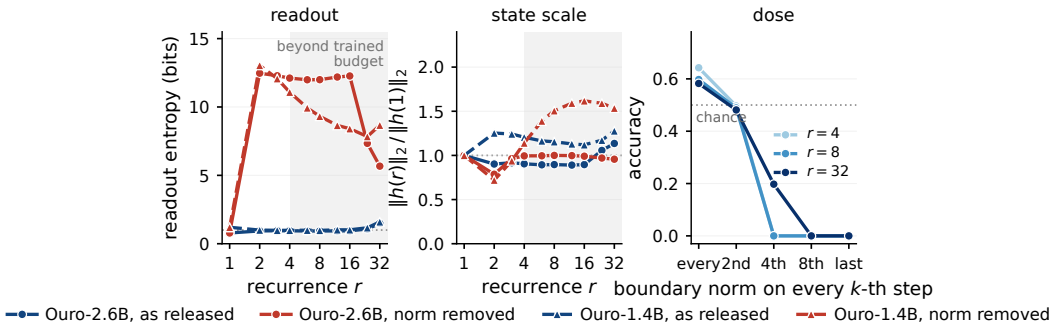

Figure 2: Mechanism on `circuit`; colours and line styles as in Fig. 1, shaded band beyond the trained budget. Left: readout entropy in bits, dotted line at the one bit of a uniform guess over two answers. Middle: state norm at r over state norm at $r = 1$. Right: Ouro-2.6B with the boundary normalisation applied every k-th loop step and always on the last, at three depths; the response is a threshold. The arms with the normalisation removed lose the readout while their state scale stays within a factor of 2.021.

The operation destroys a plateau. The unmodified models peak at the trained budget and decline slowly, from 0.642 to 0.583 on Ouro-2.6B, and the claim of this section is about what preserves that plateau. The reverse operation, adding a boundary normalisation to Huginn, is reported in Table 3 but carries no claim, since Huginn fails the competence gate on all three families before anything is done to it. The asymmetry we rely on is between removing the normalisation from released weights and training without it from scratch (Sec. 4.3).

The predictions collapse onto a single token along with the accuracy: at $r = 32$ on `hops` the unmodified arms spread their answers over 25 and 30 distinct tokens, while with the boundary normalisation removed Ouro-1.4B returns a single token for the whole split at zero mutual information with the input (App. A.7).

4.2 THE READOUT COLLAPSES; THE STATE SCALE DOES NOT

The literature on instability in looped models predicts a specific failure mode: with no boundary normalisation the residual stream grows without bound and the model fails because of the scale of the state. The prediction is not borne out on released weights. Across every cell of the grid and both arms with the operation applied, the ratio of the state norm at r to its value at $r = 1$ stays below 2.021, and the absolute norm stays in the range of the unmodified arms (Fig. 2, middle).

The readout does change, in two stages. Entropy first rises, from 0.786 bits at $r = 1$ to 12.470 bits at $r = 2$, and stays near 12.267 bits up to $r = 16$, which on a vocabulary of roughly 15 bits is close to uniform over the whole vocabulary. In this stage the zero accuracy is the arg-max of a nearly flat head landing outside a two-token alphabet (Sec. 3.4): the head has stopped answering in format. Entropy then comes back down, to 5.672 bits at $r = 32$, but not towards the correct answer: over the same cells the top-1 minus top-2 logit grows from 0.263 at the trained budget to 1.247 at $r = 32$ (Fig. 4 for both families), the modal share reaches 0.83, and accuracy stays at zero. A modal share of 0.83 with zero accuracy on a balanced binary family is possible only if the modal token lies outside the answer alphabet. The head first loses the ability to separate candidates and then becomes confident in a constant that is not an answer. Neither stage is explained by the scale of the state, which stays within a factor of 2.021 of its value at $r = 1$ over the same range.

A linear probe shows what kind of failure this is. We fit a fresh linear probe to the states of the same arm at the trained budget $r_0 = 4$ and read it on the same evaluation instances, so the two readings differ only in the head. On `circuit` the output of Ouro-2.6B with the boundary normalisation removed is zero while the probe on those states reads at chance (0.556 against a shuffled-label floor of 0.519; on Ouro-1.4B, 0.485 against 0.525), so the answer is not recoverable from the state by a linear map either. At the trained budget, then, the state no longer carries the answer linearly and the head, nearly uniform over the vocabulary, no longer answers within the alphabet; the commitment to a constant comes later, by $r = 32$. On the unmodified arm the same procedure gives 0.674 against a floor of 0.555, slightly above that arm's own output of 0.642: both readings find the answer, and the state carries at least as much of it as the trained head extracts.

The operation also admits a dose. Normalising every second, fourth or eighth loop step, always including the last so that the readout path is unchanged, orders the inference topologies by distance from the trained one, and agreement predicts accuracy that is non-increasing along that order. It is, at all 3 depths (Table 11): at the trained budget the model as released reads 0.642, every second step 0.496 (McNemar $p = 3.1 \times 10^{-9}$) and every fourth or sparser zero, with readout entropy of 0.95, 3.51 and 12.11 bits on a one-bit task. The response is a threshold, not a gradient: halving the frequency of the normalisation leaves the head at chance, anything sparser takes it to zero, out of the answer alphabet, and the state norm stays in range throughout.

The pre-registered paired test used the gap between the probe and the model output and predicted a small gap for Ouro as released and a large one after the boundary normalisation is removed. The gap moves from small to large, reversing sign, in all four pairs of arm and family, with $\Delta = -0.591$ on `circuit` for Ouro-2.6B (McNemar $p = 1.0 \times 10^{-140}$, $n = 800$), and all four survive the Holm correction (App. A.10). Since the model output after the operation is zero, the gap there equals the probe accuracy: the test measures the distance between a probe at chance and a head that has left the answer alphabet.

4.3 TRAINED FROM SCRATCH: THE NORMALISATION HURTS AWAY FROM THE TRAINED DEPTH

Two series of fifty training runs (32 on the normalisation axis, 18 on the depth axis; 16 seeds per configuration here), one on `circuit` and one on a single cell of `hops` ($d^\star = 1$, width 8; App. A.4 says why one cell). All runs are competent by the pre-declared gate, with answer-token accuracy between 0.99 and 1.00 at the end of training. The two configurations differ only in whether the state is normalised at the end of every loop step and share the trained budget $r_0 = 4$. We describe `circuit` first.

The two are closest at exactly one depth, the one they were trained at: 0.999 without boundary normalisation against 0.963 with it, $p = 0.012$ by the exact rank test over seeds (Table 4). The normalisation therefore does not prevent learning. At every other depth the gap is wider, and the

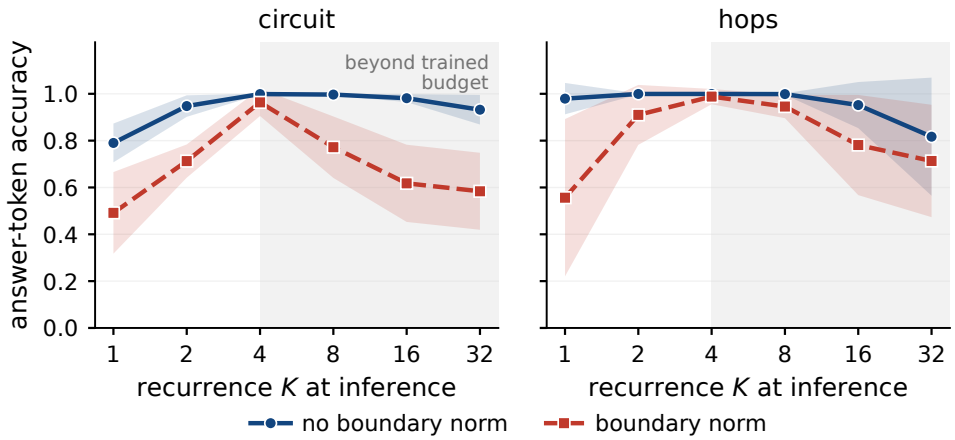

Figure 3: Models trained from scratch with the loop topology fixed before training, on `circuit` (left) and one cell of `hops` (right); means over 16 seeds, bands one standard deviation. Both configurations share $r_0 = 4$, so the comparison at equal K is free of the overshoot confound of Sec. 4.5. The curve with a boundary normalisation peaks at its trained depth and falls away on both sides; the curve without one stays above 0.948 from $K = 2$ to $K = 16$.

arm with normalisation falls away in both directions: 0.713 at half the trained depth and 0.584 at eight times it, against 0.948 and 0.932 (Fig. 3). Below the budget both arms lose accuracy and the normalised one loses more: at $K = 1$, a single application of a six-layer block, 0.492 against 0.790 ($p = 2.2 \times 10^{-7}$).

Beyond the trained budget the normalised arm falls monotonically, to 0.772, 0.618 and 0.584 at two, four and eight times the budget, while the configuration without normalisation holds 0.997, 0.981 and 0.932; the gap is widest at four times the budget. The effect is an interaction with the number of loop steps, and it separates the two sets of seeds almost completely, with Cliff's δ reaching 0.867; each of the 3 depths beyond the budget is significant on its own (p from 4.0×10^{-6} to 1.1×10^{-5}) and 3 of the 3 survive the Holm correction. The shape of the two curves is the result we rely on: the configuration with boundary normalisation peaks at the depth it was trained at and degrades on both sides of it, while the configuration without it holds above 0.981 from the trained depth to four times above it.

The second task reproduces the shape with a smaller effect beyond the budget (Fig. 3, right; Table 5). On `hops` the arms are indistinguishable at the trained depth (0.999 against 0.988, $p = 0.361$) and the normalised arm again falls away on both sides of it: 0.556 against 0.979 at $K = 1$ ($p = 2.3 \times 10^{-6}$) and 0.946 against 0.999 at twice the budget ($\delta = 0.844$, $p = 8.9 \times 10^{-6}$). The gap is still significant at four times the budget ($p = 9.0 \times 10^{-4}$) and no longer at eight ($p = 0.080$); 2 of the 3 depths beyond the budget survive Holm. The sign holds on both tasks and the size beyond the budget is task-dependent.

The direction of this effect is the main result of this half of the study. A naive reading of Sec. 4.1 would suggest that boundary normalisation is useful, because removing it breaks Ouro. Here, where the topology is fixed before training, the models that never saw normalisation inside the loop hold their accuracy over a wider range of K. Agreement explains Sec. 4.1, since a model learns a readout for the loop it has and what hurts it is a change to that loop, whatever the direction. It does not by itself explain the present section: both configurations here run with the loop they were trained with, and one still degrades beyond its budget. Agreement is therefore necessary and not sufficient, and the normalisation separates two arms that both satisfy it. The state norm points to how. The configuration without boundary normalisation lets its state norm grow across loop steps by a factor of 4.84, from 366 to 1772, and holds its accuracy almost unchanged across K; the one with normalisation holds the norm nearly constant, a factor of 1.45, and degrades. If norm growth were the cause of failure, the sign of this relation would be the other way round. A reading consistent with the sign is that growth of the residual stream damps the loop: the block's output scale is set by its internal normalisations, so as the state grows each application moves it by a smaller relative amount

and the iteration settles, whereas a boundary normalisation rescales the state before every step, so the updates keep their full relative size and a readout calibrated at r_0 sees a state that has not settled. We did not measure the per-step update size and offer this as the interpretation the norms support, in line with the fixed-point analyses cited above, not as a demonstrated mechanism.

4.4 The damage sits in the last block layer and the head

If the readout is what the operation breaks, refitting the readout against the modified loop must recover accuracy without touching the computation. That test needs weights we control, so we run it on the models of Sec. 4.3 trained with boundary normalisation, on which the same operation is available. We freeze the loop, untie the unembedding from the input embedding, and train only it and the final normalisation for 300 steps, with no gradient entering the loop. At the trained depth this recovers 33% of the accuracy lost on `circuit` (0.639 to 0.756, against 0.998 as trained, $p = 0.031$ over 6 seeds) and 31% on `hops` (0.775 to 0.845, $p = 0.062$), and stays below the model as trained on both ($p = 0.031$, $p = 0.062$; exact signed-rank over seeds). The recovered third is a plateau: 3000 steps move it from 33% to 35% on `circuit`, and the unembedding alone gives the same curve (Table 12). Unfreezing in addition the last layer of the recurrent block, shared across loop steps and holding 4.2M of the 19M parameters, returns 43% of the loss within 30 steps and 100% by 3000, and on `hops` 98% within 30. The procedure is not free: at $K = 1$, where the operation is an identity, the retrained head still gains 0.671 over 0.578, so part of the third is extra training and the plateau is the robust finding. The operation therefore damages the block's last layer together with the head that reads it, and this pair has to agree with the loop. Beyond the trained budget the single-point refit is task-dependent (Table 10): on `hops` the refitted readout matches the model as trained, while on `circuit` the model as trained has itself fallen to 0.621 by $K = 16$.

4.5 Looped against stacked depth at matched overshoot

We fix the total number of layer applications per token at 24 and split it three ways: four loop steps of six layers, two of twelve, and one of twenty-four, the last an ordinary transformer without a loop. All three are trained without boundary normalisation; the four-step loop is the no-normalisation arm of Sec. 4.3 on a separate set of six seeds. Compared at equal K the result is a tidy ordering in favour of looping, with 0.907, 0.838 and 0.490 at $K = 32$ (Fig. 5, left). That ordering is an artefact. The one-step configuration has a trained budget of $r_0 = 1$, so $K = 32$ is a thirty-two-fold overshoot for it against an eight-fold overshoot for the four-step configuration, and the left panel compares configurations that have left their budgets by very different distances. Since all three apply 24 layers at their trained budget, equal overshoot is also equal inference compute (layers $\times K = 24\,K/r_0$): the fixed-K comparison confounds overshoot with a four-fold difference in compute, and matching removes both at once. At matched overshoot the ordering is no longer detectable on `circuit` (Fig. 5, Table 6): the three configurations lie within one standard deviation across seeds of each other, at 0.907, 0.918 and 0.828 at eight times the budget, and no pairwise test rejects. This is a statement about detectability at 6 seeds: the exact Mann–Whitney test at that size rejects at the 0.05 level only for $|\delta| \geq 0.72$, and the observed $\delta = +0.33$ for the loop over the stack at eight times the budget lies inside that range.

On `hops` the confound is the same but a gap survives it and shrinks without closing (Table 7): the 4×6 loop reaches 0.957 against 0.733 for 2×12 ($\delta = 0.528$, $p = 0.180$) and 0.737 for the 1×24 stack ($\delta = 0.778$, $p = 0.026$). We apply no multiplicity correction on this axis and read it as exploratory; across the eight comparisons of Tables 6 and 7 the one significant difference would not survive Holm. Whether any advantage survives matched overshoot is thus task-dependent.

5 Discussion

We read these results as a contract between the loop and its readout: extra loop steps are usable only while the readout is calibrated for the loop it was trained with, and they stay usable further beyond the budget when the loop settles as it iterates, which a boundary normalisation prevents. The first clause is why "remove the normalisation" and "train without it" have opposite consequences; the second is why, of two configurations that each agree with their own loop, the one without the normalisation extrapolates further. The refit curve locates the contract in the last block layer and

the head that reads it. Two points follow for practice: extrapolation beyond a trained budget is best reported against the overshoot K/r_0, since the absolute axis confounds any comparison in which budgets differ; and a modification inside the loop of a released model should be assumed to invalidate its readout until measured, even when the state scale is intact.

6 LIMITATIONS

Scale and causality sit in different halves of the study: the causal half uses models of 19 to 76M parameters on two families of synthetic tasks, while the released models carry scale but cannot have their training topology varied. The effect on the normalisation axis is therefore established in sign in both halves and in magnitude only at the small scale.

Competence of the released models is limited. In the zero-shot pass only Ouro on `circuit` clears the floor; four in-context examples raise `circuit` to 0.642 and 0.619 and lift `hops` above both floors, to 0.166 and 0.154 against a majority floor of 0.072, but not to the pre-declared threshold of 0.30, which stands: `hops` enters as a secondary observation that fails the gate, and the headline result of Sec. 4.1 rests on `circuit` for both Ouro models; the collapse statement is unaffected, since it concerns the structure of the prediction and not its correctness. Huginn fails the gate on all three families (Table 3), so the direction that adds a boundary normalisation to released weights remains untested. The answer-token convention is fixed per tokeniser (Huginn merges the space and the digit into one token, Ouro separates them); an earlier convention that scored the wrong token for Huginn was caught by the in-alphabet rate, and the failure survives the correction, so it is a competence limit under our prompt and not a scoring artefact. Accuracy is scored by arg-max over the full vocabulary (Sec. 3.4); we do not report the alphabet-restricted reading, recoverable from the logged top-5 tokens, and rely on the format statistics of App. A.7 to show that the released arms after the operation have left the answer format. The causal half runs on two of the three families; App. A.4 records why `paragg` and the multi-cell `hops` grid are absent and that nothing transfers to deeper task instances.

The pre-registered primary cell was on the second evaluation tier and the few-shot pass computed the first tier only, so the paired gap test is reported on tier A at the same $r = 32$ with $n = 800$ to 900 paired instances per pair; direction and magnitude match the pre-registration, the number of cells and the sample size do not. Evaluation instances come from the materialised grid where it holds the trained cell and are otherwise generated with a disjoint index range (three of four `circuit` cells, the `hops` cell). In the widest configuration without boundary normalisation the state norm reaches 1772; our bfloat16 runs stayed stable, but we claim only that norm growth does not explain the failure, not that removing the normalisation is safe at any budget. We did not vary the readout normalisation, the axis of Sharma & Vu (2026).

7 CONCLUSION

Normalisation placement inside the loop is not a stability hyperparameter but a contract between the loop and the readout. Removing it from released weights breaks the model within a few loop steps while the state scale stays within a factor of 2.021: the head leaves the answer alphabet and later commits to a constant that is not an answer. A readout refit over a frozen loop recovers a third of the loss; retraining the last block layer with it recovers all of it. Trained from scratch, the absence of the normalisation is never worse than its presence and better away from the trained depth, although both configurations agree with their own loop: agreement is necessary for depth to be reusable, and given agreement the normalisation is a cost. At matched overshoot the advantage of looping over a stack is not detectable at six seeds on one task and reduced on the other.

REPRODUCIBILITY STATEMENT

The full code, with the protocol, task generators, verifiers, seeds and grids fixed in a single module and hashed into every logged row, is provided as supplementary material. All numbers in the text and in the tables are generated from the run ledgers by one script, and no number is typed by hand. The appendix lists the model configurations and training hyperparameters (Tables 1 and 2), the full accuracy grids (Tables 9 and 3) and every per-depth table behind the statistics in the text; the gate thresholds are in Sec. 3.4 and the arms that fail them are named in Sec. 6.

AI USE STATEMENT

Generative AI tools were used for code review and for finding bugs, and for editing this text. We did not use generative AI tools for any task with required disclosure: not for generating data, developing the conceptual framework, formulating claims, proposing or refining hypotheses, designing or giving feedback on the methodology or the experiments, implementing the methods, cleaning or reformatting data, or interpreting the results; translation and qualitative data analysis do not apply to this work. Everything not listed above was done by the authors. All experiments were run by the authors on their own hardware, and every number reported here is produced by the logged runs rather than by a language model. We reviewed all AI-assisted work: every issue raised in AI-assisted code review was checked and fixed by the authors, the protocol, the answer-scoring convention, the floors, the pre-registered test and the gates were checked by hand against the logs, and the identity check at $r = 1$ served as a standing correctness check on the implementation of the operation. We take responsibility for the final content of this work, including the text, the claims and the artifacts.

REFERENCES

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026.

Matthias M. M. Buehlmaier. LayerNorm as implicit gain control in looped transformers. *arXiv preprint arXiv:2607.10681*, 2026.

Lizhang Chen, Jonathan Li, Chen Liang, Ni Lao, and Qiang Liu. Training-free looped transformers. *arXiv preprint arXiv:2605.23872*, 2026.

Victor Conchello Vendrell, Arnau Padrés Masdemont, Niccolò Grillo, Jordi Ros-Giralt, Arash Behboodi, and Fabio Valerio Massoli. Memory-efficient looped transformer: Decoupling compute from memory in looped language models. *arXiv preprint arXiv:2605.07721*, 2026.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers. *arXiv preprint arXiv:2606.31779*, 2026.

Jacob Fein-Ashley and Paria Rashidinejad. Solve the loop: Attractor models for language and reasoning. *arXiv preprint arXiv:2605.12466*, 2026.

Zitian Gao, Yilong Chen, Yihao Xiao, Xinyu Yang, Ran Tao, Joey Zhou, and Bryan Dai. Loop the loopies! *arXiv preprint arXiv:2607.16051*, 2026.

Jonas Geiping, Sean 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. *arXiv preprint arXiv:2502.05171*, 2025.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, 2023.

Zeyi Huang, Xuehai He, Liliang Ren, Yiping Wang, Baolin Peng, Hao Cheng, Shuohang Wang, Pengcheng He, Jianfeng Gao, Yong Jae Lee, and Yelong Shen. Latent recurrent transformer: Architecture exploration, training strategies, and scaling behavior. *arXiv preprint arXiv:2605.26797*, 2026.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. LoopFormer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *International Conference on Learning Representations*, 2026.

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, and generalize: Implicit reasoning in recurrent-depth transformers. *arXiv preprint arXiv:2604.07822*, 2026.

Hsun-Yu Kuo, El Mahdi Chayti, Patrik Reizinger, Wieland Brendel, and Martin Jaggi. Stabilizing extrapolation in looped transformers via learned stochastic stopping. *arXiv preprint arXiv:2606.29983*, 2026.

Asher Labovich. Stability and generalization in looped transformers. *arXiv preprint arXiv:2604.15259*, 2026.

Wai-Lun Lam. Energy-entropy regularization: The true power of minimal looped transformers. *arXiv preprint arXiv:2601.09588*, 2026.

Joe Logan. Per-token fixed-point convergence in depth-recurrent transformers. *arXiv preprint arXiv:2607.14427*, 2026.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers. *arXiv preprint arXiv:2606.18206*, 2026.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

Costin-Andrei Oncescu, Depen Morwani, Samy Jelassi, Alexandru Meterez, Mujin Kwun, and Sham Kakade. The recurrent transformer: Greater effective depth and efficient decoding. *arXiv preprint arXiv:2604.21215*, 2026.

Andrei Cristian Popescu, Haitz Sáez de Ocáriz Borde, and Pietro Liò. Adaptive depth in looped transformers: Diagnosing learned halting gates and trajectory readouts. *arXiv preprint arXiv:2607.20519*, 2026.

Kristian Schwethelm, Daniel Rückert, and Georgios Kaissis. How much is one recurrence worth? iso-depth scaling laws for looped language models. *arXiv preprint arXiv:2604.21106*, 2026.

Rituraj Sharma and Tu Vu. Dense supervision is not enough: The readout blind spot in looped language models. *arXiv preprint arXiv:2606.24898*, 2026.

Haozhou Zhang. Chain-of-thought and compressed looped transformers: A memory-budget separation. *arXiv preprint arXiv:2605.30757*, 2026a.

Tong Zhang. When does recurrence become an algorithm? convergence selection in weight-tied looped transformers. *arXiv preprint arXiv:2607.20594*, 2026b.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A Appendix

A.1 Models

role	model	parameters	loop topology
released	Huginn-3.5B	3.5B	no boundary norm measured
released	Ouro-1.4B	1.4B	boundary norm, shared with readout
released	Ouro-2.6B	2.6B	boundary norm, shared with readout
from scratch	LoopedLM, d=512	19–76M	either, dedicated, set before training

Table 1: Models. The released checkpoints carry the scale and the models trained from scratch carry the causal claims; see Sec. 3.1 for why the two halves answer different questions.

A.2 Training hyperparameters

setting	value
width d, heads, MLP multiplier	512, 8, 4
layers $\times$ trained loop steps	6×4 (normalisation axis and loop arm); 12×2; 24×1
boundary normalisation	RMSNorm between loop steps, skipped on the last; on or off
readout normalisation	RMSNorm before the unembedding, always on
context, tokens per batch, training tokens	256, 65 536, 3×10^8
optimiser, peak learning rate, precision	AdamW, 3×10^{-4}, bfloat16
evaluation loop counts K	1, 2, 4, 8, 16, 32
seeds	16 on the normalisation axis, 6 on the depth axis

Table 2: Training hyperparameters of the models trained from scratch. All configurations share data, tokeniser, schedule and seeds; they differ only in the row that names them.

A.3 Huginn in the zero-shot pass

Huginn-3.5B	task	r=1	r=2	r=4	r=8	r=16	r=32	r=64	r=128	r=256
as released	circuit	0.004	0.000	0.000	0.311	0.414	0.448	0.445	0.453	0.451
as released	hops	0.000	0.002	0.000	0.000	0.001	0.001	0.000	0.000	0.000
as released	paragg	0.100	0.000	0.035	0.171	0.018	0.044	0.059	0.060	0.055
boundary norm added	circuit	0.004	0.000	0.000	0.299	0.416	0.446	0.452	0.446	0.450
boundary norm added	hops	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000
boundary norm added	paragg	0.100	0.000	0.039	0.203	0.023	0.050	0.054	0.051	0.053

Table 3: The reverse operation on released weights: Huginn-3.5B as released and with a boundary normalisation added, zero-shot, accuracy against the inference-time loop count on all three families. Neither arm clears the competence gate of Sec. 3.4 on any family, which is why this direction carries no claim in the text.

A.4 Scope of the causal half

The causal half runs on two of our three families, and we measured why the third is absent. A single cell of `hops` is learned by an abrupt transition at around 2750 steps, about 700 thousand examples, matching the reported shape for induction heads (Olsson et al., 2022); an eight-cell mixture trained for the same number of steps, 128 thousand examples per cell, shows no transition in any cell. Since the mixture saw fewer examples per cell than the single cell needed, these runs do not separate a per-cell threshold from a cumulative one; they establish that the grid is not learned as a mixture within our budget, which is why the causal half trains on one cell. A control task on the same data

path is learned within 250 steps, clearing the generator, the encoder, the answer position and the loss; `paragg` is not solved by any configuration.

The same models bound the claim from the other side. On the 3 `hops` cells at $d^\star = 2, 3, 4$ that no run saw in training, every configuration scores between 0.001 and 0.030 against a chance floor of 0.125, averaged over inference depths, and raising K does not help. Below the floor is the signature of Sec. 4.2 again: on an instance it has not learned the model commits to a wrong answer instead of spreading its mass. Loop depth at inference and task depth are separate axes, and our results show no transfer from the first to the second.

The control task is "copy the starting letter"; it travels the same data path, format and answer position as `hops` and is learned to accuracy one. A step is 256 sequences, and the transitioning cell is $d^\star = 1$, width 8. `paragg` remains at its floor in every configuration, including after a fix to its generator whose earlier version produced a constant label at small widths; that earlier version is the reason we verify every generator with independent code.

A.5 Normalisation axis per depth on `circuit`

configuration	K=1	K=2	K=4	K=8	K=16	K=32
no boundary norm	0.790	0.948	0.999	0.997	0.981	0.932
boundary norm	0.492	0.713	0.963	0.772	0.618	0.584
Cliff's δ	+0.94	+1.00	+0.52	+0.84	+0.87	+0.86
p (Mann–Whitney)	2.2×10^{-7}	3.3×10^{-9}	0.012	1.1×10^{-5}	4.0×10^{-6}	5.3×10^{-6}

Table 4: Models trained from scratch on `circuit`: answer-token accuracy against the inference-time number of loop steps K, averaged over 16 training seeds per configuration. Both configurations have six layers, four loop steps during training and 19M parameters. Cliff's δ and the exact Mann-Whitney p are computed over seeds, with the seed as the unit of analysis.

A.6 Normalisation axis per depth on `hops`

configuration	K=1	K=2	K=4	K=8	K=16	K=32
no boundary norm	0.979	0.999	0.999	0.999	0.952	0.817
boundary norm	0.556	0.910	0.988	0.946	0.781	0.713
Cliff's δ	+0.88	+0.53	+0.20	+0.84	+0.67	+0.37
p (Mann–Whitney)	2.3×10^{-6}	0.010	0.361	8.9×10^{-6}	9.0×10^{-4}	0.080

Table 5: Models trained from scratch on one cell of `hops` ($d^\star = 1$, width 8): answer-token accuracy against the inference-time number of loop steps K, averaged over 16 training seeds per configuration. Evaluation instances for this cell are generated held out with a disjoint index range, since the materialised evaluation grid does not contain it.

A.7 Collapse metrics

At $r = 32$ on `hops` the unmodified arms spread their answers over 25 and 30 distinct tokens, with 4.21 and 4.69 bits of mutual information between the input and the arg-max prediction and a modal share of 0.20 and 0.11. With the boundary normalisation removed, Ouro-1.4B returns a single token for the whole split at zero mutual information and a modal share of 1.000, and Ouro-2.6B returns two tokens with a modal share of 0.860. On the binary family, where a two-token prediction is not by itself a collapse, the modal share rises from 0.50 and 0.54 in the unmodified arms to 0.85 and 0.83 after the rearrangement.

A.8 TOP-1 MINUS TOP-2 LOGIT

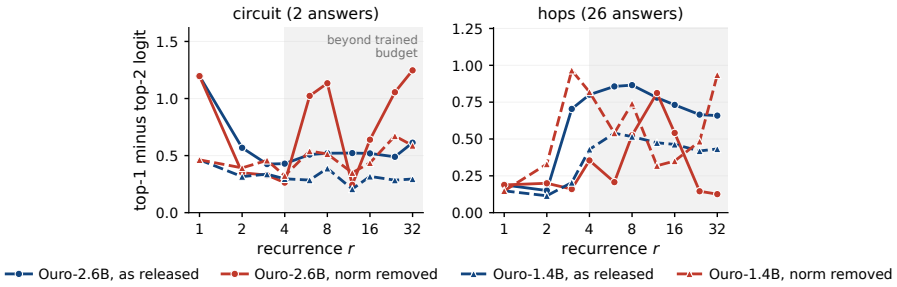

Figure 4: Top-1 minus top-2 logit against the number of loop steps for both task families. This quantity is noisy across r and is interpretable only together with the modal share; Sec. 4.2 cites it only at the trained budget and at $r = 32$.

A.9 LOOPED AGAINST STACKED DEPTH AT MATCHED OVERSHOOT

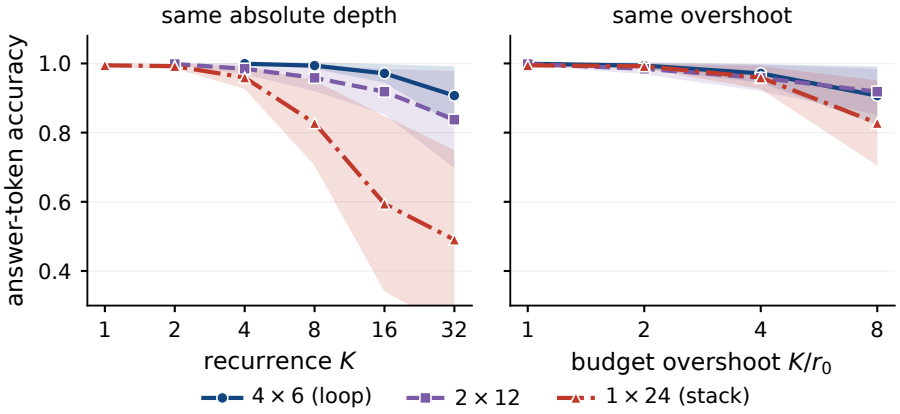

Figure 5: Three placements of a matched depth of 24 layer applications per token, trained from scratch on `circuit`. Lines are means over 6 seeds, bands one standard deviation. Left: accuracy against the absolute loop count K, which is how such comparisons are usually reported, with each configuration shown from its own trained budget upwards. Right: the same runs against the budget overshoot K/r_0, restricted to the overshoots all three configurations reach. The ordering on the left is absent on the right; that difference is the point of Sec. 4.5.

configuration	K/r_0=1	K/r_0=2	K/r_0=4	K/r_0=8
4×6 (loop)	0.999	0.994	0.971	0.907
2×12	0.998	0.985	0.958	0.918
1×24 (stack)	0.995	0.992	0.959	0.828
δ, loop vs 2×12	+0.22	+0.39	+0.11	-0.11
p, loop vs 2×12	0.589	0.310	0.818	0.818
δ, loop vs 1×24 (stack)	+0.50	+0.25	+0.25	+0.33
p, loop vs 1×24 (stack)	0.180	0.589	0.589	0.394

Table 6: Answer-token accuracy at matched budget overshoot K/r_0 for three placements of 24 layer applications per token, averaged over 6 training seeds on `circuit`. Each column compares the three configurations after leaving their trained budgets by the same factor.

configuration	K/r_0=1	K/r_0=2	K/r_0=4	K/r_0=8
4×6 (loop)	1.000	0.998	0.998	0.957
2×12	1.000	1.000	0.948	0.733
1×24 (stack)	1.000	0.997	0.925	0.737
δ, loop vs 2×12	+0.00	-0.17	+0.58	+0.53
p, loop vs 2×12	1.000	0.699	0.132	0.180
δ, loop vs 1×24 (stack)	+0.00	+0.03	+0.56	+0.78
p, loop vs 1×24 (stack)	1.000	1.000	0.132	0.026

Table 7: Looped against stacked depth on one cell of `hops`: answer-token accuracy at matched budget overshoot K/r_0 for the three placements of 24 layer applications per token, averaged over 6 seeds, with Cliff's δ and the exact Mann–Whitney p against the four-step loop. All three are trained without boundary normalisation; the four-step loop is the no-normalisation architecture of Table 5 on a separate set of six seeds, and its higher value at eight times the budget lies within the seed spread of Fig. 3. Unlike on `circuit` (Table 6), the loop retains an advantage over the stack at eight times the budget ($p = 0.026$ over 6 seeds, uncorrected), and none over 2×12 ($p = 0.180$).

A.10 Paired gap test in full

The two `circuit` pairs pass the competence gate and the two `hops` pairs do not and are reported as observations. On `circuit` the gap moves from -0.101 to $+0.490$ for Ouro-2.6B ($\Delta = -0.591$, 95% cluster bootstrap interval $[-0.609, -0.567]$, McNemar $p = 1.0 \times 10^{-140}$, $n = 800$) and from -0.011 to $+0.516$ for Ouro-1.4B; on `hops` Δ is -0.110 for both models, to three decimals.

model	task	n	gap, as released	gap, surgery	Δ [95% CI]	p
Ouro-1.4B	circuit	800	-0.011	$+0.516$	-0.527 $[-0.569, -0.490]$	$1.5 \times 10^{-126*}$
Ouro-1.4B	hops	900	-0.104	$+0.006$	-0.110 $[-0.185, -0.055]$	$4.9 \times 10^{-32*}$
Ouro-2.6B	circuit	800	-0.101	$+0.490$	-0.591 $[-0.609, -0.567]$	$1.0 \times 10^{-140*}$
Ouro-2.6B	hops	900	-0.049	$+0.061$	-0.110 $[-0.222, -0.031]$	$5.0 \times 10^{-29*}$

Table 8: The pre-registered paired test in full. The gap is the frozen probe minus the model output on the same instance; the prediction fixed before the first forward pass was a small gap for Ouro as released and a large one after the boundary normalisation is removed. Δ is the difference of gaps with a cluster bootstrap interval over (family, $d^\star$, data seed), p is the exact McNemar test on paired instances, and an asterisk marks rejection after the Holm correction.

A.11 Accuracy on the full grid

arm	task	r=1	r=2	r=3	r=4	r=6	r=8	r=12	r=16	r=24	r=32
O26-base	circuit	0.530	0.584	0.627	0.642	0.620	0.598	0.588	0.590	0.591	0.583
O26-base	hops	0.074	0.107	0.164	0.166	0.154	0.142	0.147	0.138	0.122	0.106
O26-fbraw	circuit	0.530	0.500	0.058	0.000	0.000	0.000	0.000	0.000	0.000	0.000
O26-fbraw	hops	0.074	0.071	0.049	0.082	0.038	0.000	0.000	0.000	0.000	0.000
O14-base	circuit	0.471	0.591	0.600	0.619	0.561	0.550	0.537	0.530	0.527	0.524
O14-base	hops	0.074	0.070	0.121	0.154	0.140	0.152	0.149	0.149	0.128	0.117
O14-fbraw	circuit	0.471	0.090	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
O14-fbraw	hops	0.074	0.027	0.000	0.000	0.000	0.000	0.004	0.007	0.000	0.000

Table 9: Accuracy against the number of loop steps for the released arms with four in-context examples, averaged over strata of $d^\star$ and w with $n = 100$ instances per stratum. Arms named `-fbraw` have the boundary normalisation removed. The trained budget is $r_0 = 4$.

A.12 READOUT REFIT AFTER THE OPERATION

circuit	K=1	K=2	K=4	K=8*	K=16*	K=32*
as trained	0.578	0.742	0.998	0.864	0.621	0.601
loop-boundary norm removed	0.578	0.626	0.639	0.701	0.708	0.710
+ readout refit	0.671	0.718	0.756	0.765	0.728	0.729
recovered	—	78%	33%	39%	—	—
p vs removed	0.031	0.062	0.031	0.031	0.625	0.625
p vs as trained	0.031	0.312	0.031	0.094	0.562	0.562
hops	K=1	K=2	K=4	K=8*	K=16*	K=32*
as trained	0.385	0.875	1.000	0.980	0.880	0.810
loop-boundary norm removed	0.385	0.600	0.775	0.805	0.805	0.830
+ readout refit	0.415	0.825	0.845	0.860	0.870	0.870
recovered	—	82%	31%	31%	87%	—
p vs removed	0.906	0.062	0.062	0.125	0.062	0.062
p vs as trained	0.906	0.031	0.062	0.094	1.000	0.844

Table 10: Three arms on the same loop weights of the models trained from scratch with boundary normalisation: as trained, after the normalisation is removed, and after refitting the readout against the modified loop. The refit trains only the final normalisation and the unembedding, untied from the input embedding, for 300 steps with the loop frozen. Entries are medians over the first 6 training seeds (11–66), a subset of the seeds of Table 4, so the "as trained" row is a six-seed median and differs from the sixteen-seed means there. "Recovered" is the share of the accuracy lost to the operation that the refit returns, reported where the operation costs accuracy at all. Both p values are the exact signed-rank test over seeds, paired by seed: against the arm with the normalisation removed, and against the arm as trained. An asterisk marks depths beyond the trained budget $r_0 = 4$.

A.13 DOSE: BOUNDARY NORMALISATION EVERY k LOOP STEPS

boundary norm applied	r=4	r=8*	r=32*
every step (as released)	0.642	0.598	0.583
every 2nd	0.496	0.489	0.481
every 4th	0.000	0.000	0.198
every 8th	0.000	0.000	0.000
last step only	0.000	0.000	0.000
entropy, every step (as released)	0.95	0.93	1.49
entropy, every 2nd	3.51	2.56	7.19
entropy, every 4th	12.11	11.47	13.02
entropy, every 8th	12.11	12.00	14.17
entropy, last step only	12.11	12.00	5.67
p vs as released, every 2nd	3.1×10^{-9}	1.2×10^{-5}	1.8×10^{-4}
p vs as released, every 4th	1.5×10^{-154}	1.0×10^{-143}	1.2×10^{-45}
p vs as released, every 8th	1.5×10^{-154}	1.0×10^{-143}	4.2×10^{-140}
p vs as released, last step only	1.5×10^{-154}	1.0×10^{-143}	4.2×10^{-140}
non-increasing along the dose	yes	yes	yes

Table 11: Ouro-2.6B on `circuit` with four in-context examples, the boundary normalisation applied every k-th loop step and always on the last, so that the readout path is identical across rows. Rows are ordered by distance from the trained topology. Accuracy is over $n = 800$ instances per column, eight strata of 100; p is the exact McNemar test against the model as released on the same instances, Holm-corrected within each r. Where $k \geq r$ the schedule reduces to normalising the last step only and the rows coincide exactly, and this is checked. An asterisk marks depths beyond the trained budget.

A.14 REFIT CURVE: STEPS AND SCOPE

circuit, refit steps	0	30	100	300	1000	3000
unembedding, accuracy	0.639	0.729	0.744	0.751	0.755	0.761
unembedding, recovered	0%	25%	29%	32%	33%	34%
+ final norm, accuracy	0.639	0.726	0.742	0.756	0.756	0.765
+ final norm, recovered	0%	25%	29%	33%	33%	35%
+ last loop block, accuracy	0.639	0.791	0.869	0.940	0.982	0.996
+ last loop block, recovered	0%	43%	65%	85%	96%	100%

hops, refit steps	0	30	100	300	1000	3000
unembedding, accuracy	0.775	0.845	0.855	0.845	0.870	0.865
unembedding, recovered	0%	31%	36%	31%	42%	40%
+ final norm, accuracy	0.775	0.845	0.860	0.845	0.870	0.860
+ final norm, recovered	0%	31%	38%	31%	42%	38%
+ last loop block, accuracy	0.775	0.995	1.000	1.000	1.000	1.000
+ last loop block, recovered	0%	98%	100%	100%	100%	100%

Table 12: Recovery of the accuracy lost to the operation, by number of refit steps and by what is unfrozen: the unembedding alone; plus the final normalisation; plus the last loop block, i.e. the last layer of the recurrent block, shared across loop steps (4.2M parameters). Loop weights below it stay frozen throughout. Medians over the first 6 seeds at the trained depth; "recovered" is the share of the loss returned, computed from unrounded medians. Step 0 is the untouched head and coincides with the arm after the operation by construction, checked per seed. The readout alone plateaus at a third on both tasks; with the last block the loss is recovered in full.