The Loop Records Where It Slipped: Verifying Execution Slips in Looped Language Models

Anonymous authors
Paper under double-blind review

Abstract

Looped language models expose a hidden state after each iteration at every token. We ask whether these states identify execution slips: wrong digits in otherwise largely correct arithmetic traces. On synthetic multi-digit addition with Ouro-2.6B, a linear probe localizes a wrong digit in 97.2% of eligible erroneous answers, and error evidence becomes more readable across iterations of the same weights. Used as a verifier with a simple arithmetic length check, it selects the correct answer in 95.6% of eligible completed problems from four sampled candidates, versus 89.7% for majority voting under the same filtering. On the 253-problem retry set, selective resampling reaches 92.5% at about $1.2\times$ the nominal generation budget, from 87.0% greedy accuracy. State transplantation gives partial, donor-dependent digit correction; removing the probe direction yields none. Ouro-1.4B and Huginn provide weaker replications; negative results on GSM-hard and knowledge tasks show that useful verification requires both discriminative scores and correct alternatives. Disagreement between recomputations can train the detector without gold labels, although selection uses gold-supervised probes. In a four-step addition chain, a fixed probe gate improves end-to-end accuracy from 44.2% to 58.3%.

1 Introduction

A looped language model runs one shared block T times per token, so the number of iterations is a compute knob set at inference (Saunshi et al., 2025; Geiping et al., 2025; Zhu et al., 2025). The model still writes out a chain of thought, and a text-based verifier sees only that trace. But the loop leaves something else behind: a hidden state after every iteration, at every token. This paper asks what those states say about the correctness of the digit being written (Figure 1).

We look at a narrow but clean failure, which we call an execution slip. In multi-digit addition, Ouro-2.6B often writes the column-by-column work correctly and then emits a final number that does not match its own columns. Resampling alone does not cure this, because different samples slip on different digits. In our verifier set the right answer is almost always among four samples, yet majority voting misses it about one time in ten. The problem is recognizing the correct candidate that is already there.

Our tool is a logistic-regression probe on the hidden state that writes each answer digit, trained to say whether that digit came out right. We call it SLIP, for Selective Loop-Internal Probe. It localizes wrong digits and gets much sharper as the loop iterates. Used to score candidates, it beats majority voting, confidence-based selectors, two process reward models, and a rule that checks the written columns against the answer. Ablating the probe direction leaves the output unchanged. Patching from a correct run sometimes repairs a digit, but provides only partial, donor-dependent correction rather than a general repair policy. We therefore use the signal to check fresh computations.

This is also where the method breaks down. Selection helps when the generator produces correct alternatives and the probe can tell them apart. Synthetic addition satisfies both conditions. On GSM-hard the errors are detectable, but the candidate generators we tried rarely produce a better answer, so there is little to select. On multiple-choice knowledge tasks the answer log-probability is already the better signal. We also find that a detector can be trained from disagreement between two recomputations, with no gold answers at all.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

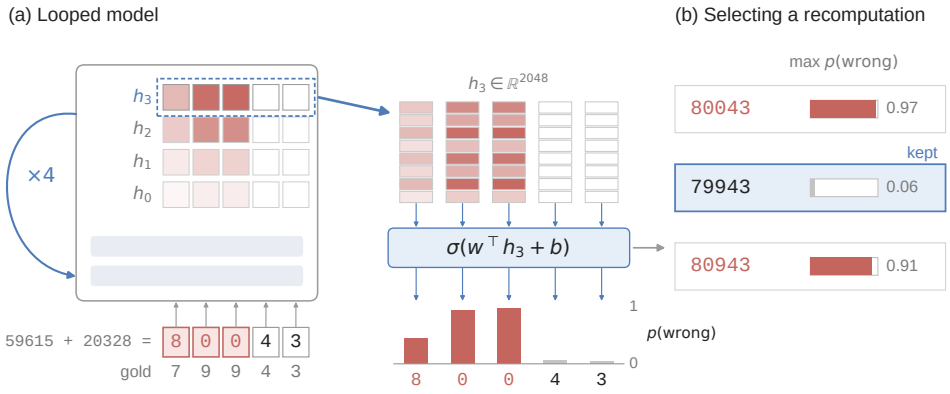

Figure 1: Overview on a recorded error ($59615 + 20328 \rightarrow 80043$; gold 79943; the first three digits are wrong). (a) Ouro applies one shared block for four loop steps to every token, leaving a hidden state per answer digit after each step (h_0–h_3); evidence for the wrong digits builds toward the final step. That final-step state is pulled out and read by a logistic-regression probe that gives each digit a correctness score q_j, drawn here as the slip score $1 - q_j$ so that the highest bar marks the slipped digit. (b) The same score verifies recomputed candidates, keeping the one whose worst digit scores highest. Scores and shading are schematic; measured scores for this example, including a missed leading digit, are in Figure 2.

1. **Localization, and how it forms across the loop.** On several thousand Ouro-2.6B generations, SLIP points to a wrong emitted digit in 97% of erroneous answers, and problem-difficulty features do not explain this. With the same weights and the same text, more loop iterations expose more decodable evidence. Ouro-1.4B and Huginn replicate the pattern with different strength.
2. **Selection from small candidate pools.** On synthetic addition, a probe trained on the candidate distribution recovers over four fifths of the oracle headroom over greedy decoding from four or five candidates. It beats majority voting at both pool sizes, and two process reward models and a text-consistency rule on identical candidates. Flagging a fifth of the problems for one extra sample captures most of this gain at a small fraction of the cost. A *detector* can also be trained from disagreement between recomputations, without gold answers; selection is not.
3. **Detection, intervention and recovery come apart.** Ablation yields no correction and patching gives only partial donor-dependent correction; strong detection on GSM-hard does not turn into a retry gain. Readable error evidence and available correct candidates are separate requirements. Where both hold, gating each step of a four-step addition chain with the probe lifts end-to-end accuracy substantially, a first test of internal verification inside a longer computation.

2 Related Work

Internal error detection. Sun et al. (2025) decode arithmetic outputs and correct answers from the activations of a standard transformer and re-prompt the flagged reasoning steps. Orgad et al. (2025) study how errors are represented internally. Kumaran et al. (2026) find post-answer confidence signals that predict correctability and can restore detection under corruption. Goel et al. (2026) separate detecting a corrupted context from predicting the final answer, and report that intervention outcomes depend on the model and the error type. Yuan et al. (2026) likewise find strong detection with unreliable steering and patching. We study the same detection–intervention gap at the level of single emitted digits in a looped model, and put the detector to work for candidate selection.

Looped and recurrent-depth models. Looped models scale computation by applying shared parameters repeatedly (Saunshi et al., 2025; Geiping et al., 2025; Zhu et al., 2025). Recent work examines their dynamics, readouts, and serving (Popescu et al., 2026; Yang et al., 2026; Sharma & Vu, 2026; Blayney et al., 2026; Schwethelm et al., 2026). Closest to us, Kirin (2026) probe process

quality in frozen Ouro, study how readability depends on recurrence, replicate parts of the analysis on Huginn, and obtain selective-prediction and terminal-selection gains, including on well-formed candidate pools, while the tested generative interventions fail. We do not claim to be first to show internal monitoring or recurrence-dependent readability in these models. What we add is digit-level localization, verification with distribution-matched probes on matched candidate pools, and detector training from disagreement; our iteration curves describe how the decodability of a digit error builds up over steps.

Reasoning verification. Process reward models score written reasoning steps (Lightman et al., 2024; Wang et al., 2024), ProcessBench evaluates step-error localization (Zheng et al., 2025), and outcome verifiers score final correctness (Cobbe et al., 2021). In about seven of ten synthetic answers with a wrong emitted digit, the written column digits reassemble to the correct answer. A rule that checks for this mismatch catches most slips but has enough false positives to limit selection, and the two 7B PRMs we tried select less accurately than SLIP. This is a limitation of the two PRMs we tested, and does not mean that the written reasoning lacks error information: the column-mismatch rule already extracts it. The carry-like errors we observe connect to mechanistic accounts of transformer arithmetic (Quirke & Barez, 2024; Nikankin et al., 2025).

Test-time selection. Self-consistency (Wang et al., 2023), agreement-based early stopping (Li et al., 2024), and confidence weighting (Fu et al., 2026) choose among generations from output statistics. On small fixed pools the probe recovers more of the available headroom. On BBH object counting, argmax loses to voting; an over-scored wrong candidate can defeat an imperfect verifier (Stroebl et al., 2026), and voting itself can be non-monotone in K (Chen et al., 2024). Our synthetic K=16 argmax failure instead largely reflects a digit-label blind spot, addressed by an arithmetic length check. Our label-free detector distills disagreement between recomputations into a single forward pass; related ideas use logical-consistency constraints (Burns et al., 2023) or semantic entropy over samples (Farquhar et al., 2024).

3 Setup

Models. Ouro-2.6B (our primary model) and Ouro-1.4B (Zhu et al., 2025) iterate a shared stack of 48 (resp. 24) layers with hidden size 2048 T=4 times per token, exposing the hidden state after each loop step, $h_0, \ldots, h_3$; T can be changed at inference. The second recurrent architecture is Huginn-0125 (Geiping et al., 2025), a 3.5B recurrent-depth model whose recurrence count r is set at inference (we use $r \in \{1, 8, 16, 32\}$) and which, unlike Ouro, is trained without per-iteration supervision. The non-looped control is Qwen3-1.7B-Base (Qwen Team, 2025), whose hidden states we probe at every layer. All models are frozen.

Tasks. (i) Synthetic multi-digit addition with operand length $d \in [5, 10]$ and chain-of-thought prompting, a high-error, execution-dominated task. On 2.6B we have 3,578 problems: an original 469-problem set plus a 3,109-problem expansion generated with a disjoint seed. Results on the original set are marked as such; the subsets are listed in Appendix A. (ii) GSM-hard (Gao et al., 2023), GSM8K word problems with larger numbers substituted in. (iii) Auxiliary tasks: median finding and character counting.

SLIP. For each digit of the final answer we take the hidden state at each loop step at the position that predicts it, obtained from one teacher-forced pass, and train a logistic-regression probe (Alain & Bengio, 2016) per loop step. Gold-supervised probes label each emitted digit by comparing it with the gold digit at the same position; Section 5.1 covers supervision from disagreement. The target is a *correct* emitted digit, so $q_j = \hat{P}(\text{digit } j \text{ is correct})$ and the answer score is the bottleneck $s = \min_j q_j$: low scores trigger a retry and high scores win candidate selection. Figures 1 and 2 plot the complementary slip score $1 - q_j$, so that the highest bar marks the suspected digit. Evaluation uses problem-grouped folds. Digits are matched to gold from the left, which leaves one blind spot: an answer that is a strict prefix of gold has every emitted digit labelled correct and would therefore score $s \approx 1$. For synthetic candidate selection and 2.6B fixed-fraction retry we reject an answer shorter than the longer positive operand, without consulting gold. This check does not catch a d-digit prefix of a $(d{+}1)$-digit sum when both operands have d digits. It is applied to every candidate, including greedy, and to all selection baselines. Appendix A reports what the check is worth. Generations that cannot be aligned to the text are excluded from probe data; selection accuracy is exact answer correctness on the stated eligible sets (Appendix A).

Table 1: Slip detection across models, at token and problem level (AUROC); 95% bootstrap CIs over problems (2.6B: 3,578 problems, 506 with a wrong emitted digit, length errors included; 79 of the 3,578 completed generations emit only a shortened prefix of gold, contain no wrong digit, and so have every digit labelled correct under this token-level target; 1.4B: 888/119; Huginn: 272/128, few-shot, §6). Last column: token AUROC at the first and last loop step. The T=1 row is the dose–response run on the 258-problem greedy set (§4), where the same weights at T=4 read 0.976.

	token AUROC	problem AUROC	first→last step or layer
Ouro-2.6B (loop step 3)	0.985 [0.979, 0.990]	0.991	0.848 → 0.985
Ouro-1.4B (loop step 3)	0.965 [0.937, 0.982]	0.925	0.796 → 0.965
Huginn-0125 (recurrence 16)	0.847 [0.810, 0.880]	0.787	0.810 → 0.847
Ouro-2.6B, T=1 (258-problem set)	0.821 [0.742, 0.877]	—	—
Qwen3-1.7B (best layer = final)	0.837 [0.793, 0.879]	0.769	0.471 → 0.837

4 A Token-Level Slip Signal that Forms Across the Loop

Table 1 and Figure 2 give the detection results. On Ouro-2.6B the last-step probe separates wrong from correct digits at 0.985 token AUROC, and the minimum token score per problem gives 0.991 at the problem level. The per-step curve is the part we find most interesting. The first loop step already carries some evidence (0.848), the second adds little (0.876), and the third jumps to 0.982. Concatenating features from several steps adds at most half a point (Appendix B).

What the probe reads. A probe could score well simply by recognizing hard problems, so we checked localization. Restricted to wrong answers, the probe still separates wrong from correct digits, and its single lowest-scored token is a genuinely wrong digit in 97.2% of erroneous answers (excluding the few in which every digit is wrong). A sequence-level score such as $p(\text{True})$, the model's own estimate that its answer is right, cannot do this. Localization is where model size shows. Ouro-1.4B detects almost as well as 2.6B (Table 1), but its top-1 falls to about four in five: its state records that a slip happened and roughly where, but the probe more often lands on a neighbouring digit. A baseline over problem-observable features (answer length, digit position, operand digits, carries) reaches only 0.695 token AUROC on the merged set, and its lowest-scored token is a wrong digit only 13% of the time, against the probe's 97% (Appendix A). Carry structure is predictable from the operands; where the model actually slips is not. Nor is the signal simply the hidden-state norm: unit-normalizing features leaves AUROC unchanged. This control is motivated by the observation that per-loop language-model supervision need not control hidden-state scale (Sharma & Vu, 2026).

Further controls (Table 4). Detection survives restriction to the first wrong digit; an MLP ties the linear probe and trajectory features add half a point with overlapping CIs. A separately trained probe on gold-defined pre-error positions reaches 0.983 problem AUROC, versus 0.834 for problem-observable difficulty features, and remains strong after matching operand length and carry count. This supports predictive information beyond those features, but does not exclude learned difficulty or establish an online stopping policy: the window uses gold and excludes first-digit errors. Appendix B gives the stratified and leading-digit analyses.

Error census and interventions. Among length-preserving wrong answers in the expansion set, 44% differ from gold by exactly ±1 in every mismatched digit, which is what a carry error looks like from the outside; spurious carries far outnumber dropped ones, and the remainder are other digit substitutions. These are output-level categories and do not identify the underlying computation. Decodability rises between the second and third loop step for carry errors and non-carry errors alike (Figure 2b). We tried two interventions. Patching a correct run's loop state into the wrong run corrects the slipped digit in 68/345 cases (19.7%). On the 158 problems with both donor conditions, correct-run and wrong-run donors correct 29 and 19 digits, respectively. A wrong run can still supply a locally correct digit; its 36 output changes all match the donor digit. This supports partial donor-digit transfer, rather than a general repair policy (Appendix B). Projecting the probe direction out of the forward pass blinds the probe completely and changes the emitted digit in none of the problems tested. Two further tests also show limited direct correction: re-decoding from the flagged digit almost never repairs the answer, and a classifier that picks among the ten possible digit values recovers the true digit only 0.159 of the time (§5). We therefore treat the signal as diagnostic and spend compute on verifying regenerated answers. One further observation fits a sequential-carry picture: on 2.6B,

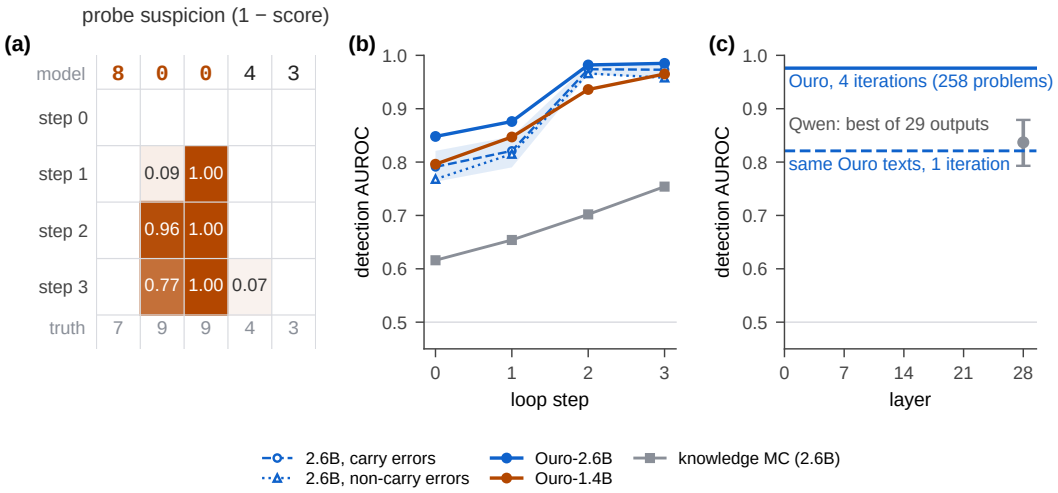

Figure 2: (a) A real slip ($59615 + 20328 \rightarrow 80043$; correct 79943): per-digit probe suspicion $1 - q_j$ by loop step: little signal at step 0; the slipped digits light up as the loop iterates (the leading digit escapes detection, a genuine miss). (b) Detection AUROC by loop step: evidence accumulates on arithmetic (carry and non-carry errors alike) and on knowledge tasks. (c) Controls: the best of 29 Qwen3-1.7B outputs is shown (0.837, with 95% CI). On the 258-problem fixed-text Ouro set, $T = 1$ and $T = 4$ give 0.821 and 0.976; Qwen uses its own generated answers on 300 problems. Type-specific curves in (b) measure localization within erroneous answers, while the model-wide curves include no-slip answers.

accuracy on 10-digit additions jumps by 30 points from T=3 to T=4. This did not replicate on 1.4B (§6).

Dependence on the iterations. With identical weights and generated texts, forcing T=1, 2, 3, 4 gives last-step AUROC $0.821 \rightarrow 0.838 \rightarrow 0.966 \rightarrow 0.976$ on the 258-problem greedy set, with the same shape on the greedy+boost union (Appendix A). Since Ouro's step k does not depend on the eventual loop count, these are earlier readouts of the same per-token computation, not an independent replication of the step curve. For a non-looped control, we swept all 29 Qwen3-1.7B hidden-state outputs, since error signals need not peak at the final layer (Orgad et al., 2025). Its best layer gives 0.837, below both Ouro sizes, but sequential depth is unmatched: four Ouro-2.6B iterations use 192 layer evaluations versus Qwen3's 28. Ouro at T=1 uses 48 layers and reads 0.821 on different problems, so this is not a paired architectural comparison. Finally, a probe on the full output-logit vector reaches 0.979 on the same tokens; ten digit logits give 0.799, and entropy or maximum probability 0.818. The signal survives into the full logits but is weaker in compact confidence summaries. These controls establish increasing readability with iterated computation, without separating recurrence from its additional compute or training.

5 SLIP as a Verifier: Turning Failed Samples into Accuracy

The right answer is usually in the pool (any-of-4 is 0.972 on average). Selectors use the same alignable candidate members and the same arithmetic length check, including for greedy when it is added (Appendix A). The nominal budgets are $4\times$ for four samples, $5\times$ with greedy added, and about $1.2\times$ when only the lowest-scored 20% of greedy answers receive one sample. For retry, both answers are scored and the better is retained. Probes use problem-level folds, so a scored problem never appears in their training data.

Candidate selection (Table 2, Figure 3). SLIP averages 0.956 over three seeds against 0.897 for majority voting on the same filtered candidates (paired problem-clustered 95% CI for the gain $[+4.0, +8.0]$ points). Adding greedy gives 0.967, from greedy 0.869 toward the any-of-5 ceiling 0.980. Majority voting with greedy reaches 0.937; SLIP leads by 3.0 points $[+1.6, +4.5]$. Ouro-

Table 2: Verifier comparison on multi-digit addition, Ouro-2.6B. **Left:** selectors use the same alignable samples and arithmetic length check on the 253-problem seed-1 set; scoring overheads differ. **Right:** nominal generation budgets relative to one depth-4 generation, excluding scoring and training. Four-/five-candidate results average seeds with $n = 253/252/253$; K=16 uses 253 problems and one seed. ‡Earlier baselines use 258 problems without the check; SC K=8 averages three seeds. †The 20% retry fraction was fixed after exploration. DeepConf is a small-K offline reduction.

Selector	Signal	Acc. (seed 1)
(greedy, 1×)	—	0.870
Majority vote (SC)	frequency	0.885
Weighted vote	+logprob	0.889
DeepConf-offline	conf. filter	0.779
Math-Shepherd PRM, 7B	written steps	0.834
Qwen2.5-Math-PRM, 7B	written steps	0.901
Page–answer rule	written columns	0.909
SLIP (ours)	loop hidden states	**0.953**
Oracle (answer present)	—	0.964

Strategy	Cost	Acc.
Uniform depth 5	1.25×	0.830
Re-prompting‡	2×	0.868
ESC (w=2)‡	2.9×	0.860
Self-Refine‡	3×	0.857
SC maj., K=4+greedy	5×	0.937
SC maj., K=8‡	8×	0.947
SC maj., K=16	16×	0.984
SLIP (ours)		
Temp. retry†	1.2×	0.925
5 cand. (SC+g)	5×	0.967
16 cand. (argmax)	16×	**0.988**

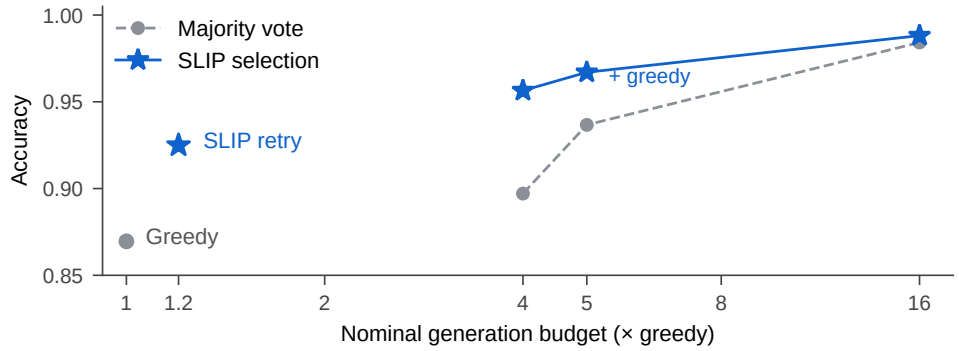

Figure 3: Accuracy versus nominal generation budget (log x), Ouro-2.6B. Candidate selectors use the same alignable members and length check. The 4–5× points average three seeds; 16× uses one seed. The 1.2× point retries a fixed 20% within each fold. Scoring and training costs are excluded.

1.4B also gains with greedy available ($0.865 \rightarrow 0.917$), although selecting among its four sampled candidates alone does not beat greedy.

Selective retry. Flagging the lowest-scored 20% within each evaluation fold, drawing one temperature-0.7 sample, and keeping the probe's choice gives 0.925 averaged over seeds, from greedy 0.870 (gain +5.5 points, problem-clustered CI $[+3.3, +8.0]$). The three accuracies are $0.929/0.925/0.921$ (Table 10). The fraction was fixed after exploration, so this is a fixed-budget result. The earlier adaptive-fraction analysis without the length check gives 0.916 under corrected nested cross-validation; the 1.4B fixed-fraction result without the check is 0.886 (Appendix C).

Other selectors. On seed 1, log-probability weighting is close to majority voting, while the small-K confidence filter discards useful candidates. Across seeds, Math-Shepherd (Wang et al., 2024) selects below majority; Qwen2.5-Math-PRM (Zhang et al., 2025) reaches 0.917, 4.0 points below SLIP $[+2.4, +5.7]$. The page–answer rule reaches 0.913 with four candidates and 0.925 with greedy, versus SLIP's 0.956 and 0.967 (differences +4.4 $[+2.8, +6.2]$ and +4.2 $[+2.6, +5.9]$). All use the same length check. Written text therefore carries useful error information; the internal readout selects more accurately in these tests.

Larger K. At K=16, SLIP reaches 0.988 versus 0.984 for majority on the same filtered pool: the paired difference +0.4 points $[-1.6, +2.4]$ does not establish superiority. Without the length check, argmax reaches only 0.913, largely because short prefix-only errors receive positive digit labels. The

check removes this collapse without changing aggregators (Appendix A). Low token calibration error alone does not prevent candidate-level mistakes (Appendix B).

Training distribution. A greedy-only probe is out of distribution on sampled candidates. On the original 1.4B set, its pooled candidate-correctness AUROC falls to 0.691 and four-candidate selection falls below majority; adding sampled training generations restores AUROC to 0.872. An equal-token control retains a gap above ten points (Table 11). On 2.6B the effect is weaker: greedy-only selection already beats majority, the equal-token gap is about one point, and the mixed-training selection gain has an interval crossing zero. Distribution matching is therefore a useful check, not a universal requirement. These diagnostics predate the arithmetic length check, and their AUROCs pool candidates across problems rather than measuring within-problem rankings.

Repair versus recomputation. Temperature sampling and depth variation both produce useful alternatives; nothing in the results depends on depth specifically. Restarting only the answer does not work. Re-decoding from a flagged digit at other depths or temperatures repairs $1/50$ tested errors, and a ten-way classifier on the same states, which decodes the gold digit at 0.996 on already-correct positions, reaches 0.159 on slipped ones. This low score is a limitation of our decoder and does not prove the true digit is missing from the state. It does contrast with the successful answer decoding of Sun et al. (2025) in a three-digit setting, and shows that how useful a readout is depends heavily on the kind of error it has to recover. Text-level repair also gives no net gain. On the separate 258-problem set, unconditional answer-level re-prompting (a simplified form of Sun et al. 2025, which re-prompts at the erroneous step) fixes none of the 33 initially wrong answers and breaks one correct answer ($224/258$, or 0.868). One-round Self-Refine (Madaan et al., 2023) fixes one and breaks five ($221/258$, or 0.857; cf. Huang et al., 2024). Changing the computation path, by temperature or by depth, produced better alternatives in these tests than editing the text did.

5.1 Label-free training and selective prediction

Disagreement as supervision. Generate each problem at two depths (T=$4/5$ on 2.6B, T=$4/3$ on 1.4B) and mark a digit suspect where the two answers disagree. The probe is then fitted in the same direction as the supervised one, with agreement as the positive class, so its scores are read exactly like q_j. These pseudo-labels are noisy (Table 3), yet a probe trained on them alone reaches 0.893 token AUROC, against 0.977 with gold labels on the same evaluation set. The recipe transfers to 1.4B and, more weakly, to Huginn (Table 3). The extra cost is one additional generation per training problem, paid once; deployment is the same teacher-forced rescoring pass as for the supervised probe. On GSM-hard we use a larger teacher pool, digit agreement across eight alternative generations, and obtain 0.773 problem AUROC without gold labels; the gold-supervised exact-answer probe on the same task reaches 0.830 (Appendix D), under a different evaluation protocol, so we read the two only as the same order of magnitude. The label-free value is below the six-sample agreement baseline (Table 6) but needs no alternatives at deployment. Zero-shot transfer of a probe from addition to GSM-hard is weak (0.645), so this is a per-task recipe. It gives label-free detection; label-free selection remains untested.

Selective prediction. Ordering GSM-hard problems by probe score gives an area under the risk–coverage curve (AURC) of 0.077, against 0.223 when unordered; at half coverage the retained answers are right far more often than the base rate (Appendix D). Even where retry has no headroom, the probe works as an abstention signal.

5.2 Headroom on GSM-hard

On GSM-hard, end-to-end retry does not reliably help: every configuration we tried lands within about a point of greedy. The constraint is headroom. An oracle over depths $\{4, 5, 7\}$ gains only +3.8pt here, against +14–28pt on synthetic addition, so there is little for any policy over these candidates to find; other generators or prompts might do better. Selection is no better even with a matched probe (0.730 against greedy 0.823): the probe's member discrimination is moderate, the any-of-7 ceiling over greedy and six T=0.7 samples sits only a few points above greedy, and at that precision global swaps lose more correct answers than they rescue. GSM8K marks the other boundary: with far fewer residual errors, only 4% of them are slips, against 31% on GSM-hard, and detection is correspondingly weak. On GSM-hard the probe still beats the answer-logprob baseline without an

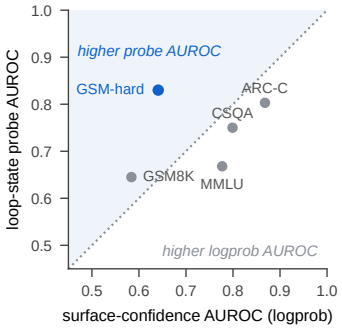

Figure 4: Detection context across tasks: answer-logprob AUROC versus problem-level probe AUROC. GSM points use different eligible subsets and are descriptive, unpaired comparisons; knowledge points use per-task probes. Arithmetic is omitted because its existing scores use different correctness targets. The diagonal marks equal AUROC, not a deployment boundary.

extra generation and gives per-digit scores; SC agreement is stronger but costs five extra generations per problem (Table 6).

5.3 Gating a chained computation

When four additions are chained, fewer than half of the chains end correct (Table 8). Each step is its own prompt and only the flagged step is redrawn, with the number it produces passed on, so a repair costs one step rather than the rest of the chain. Gating each step (flag, resample once, let the probe choose) lifts this by 14 points at about $1.2\times$ cost (paired $[+9.4, +19.2]$), a random gate set to the probe's flag rate gains almost nothing ($+1.9$, $[-0.3, +4.4]$) while in fact firing rather more often, and an oracle gate shows that the probe recovers about 70% of the available headroom. The probe is trained on single-addition data and its threshold is set on that training pool; both are frozen before any chain is evaluated, making this a small transfer test (Appendix A). This is one pipeline type, tested in isolation.

6 Generality Across Tasks and Models

The numbers behind this section are collected in Table 8.

Other tasks. Probes trained in-task find signal on addition, on the auxiliary median and counting tasks, and on GSM-hard exact-answer detection, but the probe weights do not transfer: an addition-trained probe falls to little better than chance on the other tasks, a harder shift than the within-arithmetic transfer of Sun et al. (2025). We claim a per-task probe that a few hundred problems can train, and no more. The headroom account carries over: uniform depth changes are flat or harmful on the auxiliary tasks, while per-problem oracle selection would gain several points, so selection is the right shape of intervention there too.

Knowledge tasks. On multiple-choice problems from MMLU, ARC-Challenge and CommonsenseQA (Hendrycks et al., 2021; Clark et al., 2018; Talmor et al., 2019), the answer-letter position shows the same rising curve across loop steps, but the practical picture inverts (Figure 4): answer log-probability is the better detector, and adding the probe to it gains nothing. Label-free distillation still works there. The comparison cuts both ways: internal scores beat compact confidence features on arithmetic and lose to them on knowledge questions.

BBH. Before running two BBH subtasks (Suzgun et al., 2023) we recorded a prediction from an error census: object counting, where most errors are off by one, should suit internal verification, and multistep arithmetic, with no near-misses at all, should not. The detection half held in direction but not in size. Pooled across problems, candidate-level discrimination is at chance on multistep arithmetic and well above it on object counting. Ranking candidates within a problem, which is what selection needs, the object-counting signal survives in every seed but shrinks to a little over chance, and on multistep arithmetic too few problems have both a right and a wrong candidate to measure anything (Appendix C). The predicted selection advantage did not hold. Four-candidate argmax loses to majority on object counting; probe-weighted voting over four samples plus greedy reaches 0.861, matching majority over the same five candidates. The weak within-problem signal therefore does not establish a selection advantage over voting (Appendix C).

Transcription errors. A fidelity suite (copying long numbers, reversing digit strings, tracking variables), prompted to produce nontrivial error rates, gives a clear negative: reversal and tracking errors

are detected only weakly. Whatever the arithmetic probe reads, it is not a generic transcription-error signal.

Other models. Zero-shot, Huginn-0125 (Geiping et al., 2025) almost never solves our additions, which is useless for a slip study. Two few-shot exemplars bring it to about half, a third of its errors are then near-miss slips, and the signal appears (Table 1), rising with recurrence count with a paired interval that excludes zero. Huginn knowledge detection stays near chance across recurrence counts, unlike Ouro's rising curve; neither supplies a practical advantage over confidence on those tasks. The verifier transfers in a weakened form: temperature sampling wrecks few-shot accuracy, but candidates drawn at different recurrence counts keep it, and probe-weighted voting over them gains about eight points. Huginn, unlike Ouro, is trained without per-iteration supervision and its step curve is much shallower. We suspect the sharp step-wise formation on Ouro is a signature of that supervision, but have not tested this.

Ouro-1.4B, the smaller model, replicates the rising detection curve, the five-candidate gain and the fixed-20% retry gain (Tables 1 and 10), though SC-only reranking does not beat greedy. The one exception is the depth-scaling curve of §4: 1.4B is non-monotone in depth (cf. Yang et al. 2026). Depths beyond the training value $T{=}4$ are out of distribution; generation is near-complete at depths 5–8, although accuracy can decline.

7 Limitations and Conclusion

The caveats scattered through the paper come down to a few facts. The strong results are on synthetic addition; on GSM-hard the probe detects but cannot select, and on knowledge tasks log-probability is the better signal. Two Ouro sizes, a weaker Huginn replication and one non-looped control do not separate recurrence from training, architecture, or total compute, and the full-logit result shows that the evidence is not confined to hidden states. The token-level target does not cover answers that are truncated prefixes of gold; we reject some of these with an arithmetic length check at scoring time, while the remaining prefix errors stay outside the token target (Appendix A), and some incomplete or unalignable generations are excluded from probe data. The label-free result is about detection; every selection gain uses gold-supervised probes. Costs are nominal generation budgets that leave out probe-data collection and each verifier's scoring pass. Several evaluation sets of different sizes reflect the history of the project and are listed in Appendix A; the adaptive-retry analyses and exploratory aggregators have the limits described in Appendix C. Finally, a calculator would verify these additions. We use addition because its errors are frequent, small, and checkable, which makes it a good place to study what a looped model's state records about its own mistakes.

Conclusion. The loop keeps enough information to locate a wrong emitted digit, and that information becomes more linearly readable as the iterations proceed. On small candidate pools, the readout with a shared length check selects more accurately than voting at the same generation budget. State transplantation gives partial donor-dependent correction, while probe-direction ablation yields none. The verifier cannot supply an answer that was never in the pool. It works when the probe was trained on the kind of candidates it scores, when the generator actually produces better alternatives, and when the scores can tell those apart. Where gold labels are missing, disagreement between two recomputations is enough to train the detector. The chained-addition result suggests where this leads: a model that checks each step of a long computation against its own intermediate states and resamples only the steps that fail the check, so that a slip is stopped where it occurs instead of being carried into everything that follows.

AI use statement

In this work, we used generative AI tools for designing experiments, generating the synthetic addition problems, implementing methods and analysis code, data cleaning, and interpreting results. We have not used generative AI tools for proposing hypotheses, developing theoretical models, formulating mathematical claims, or writing proofs. Additionally, we used generative AI tools for drafting and editing the text, producing figures, and literature search. We reviewed AI-assisted code, checked reported results against saved artifacts and evaluation scripts, and checked AI-suggested references against the cited sources. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016. doi: 10.48550/arXiv.1610.01644. URL https://arxiv.org/abs/1610.01644.

Hugh Blayney, Álvaro Arroyo, Johan Obando-Ceron, Pablo Samuel Castro, Aaron Courville, Michael M. Bronstein, and Xiaowen Dong. A mechanistic analysis of looped reasoning language models. *arXiv preprint arXiv:2604.11791*, 2026. doi: 10.48550/arXiv.2604.11791. URL https://arxiv.org/abs/2604.11791.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *ICLR*, 2023. URL https://openreview.net/pdf?id=ETKGuby0hcs.

Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more LLM calls all you need? Towards the scaling properties of compound AI systems. In *NeurIPS*, volume 37, pp. 45767–45790, 2024. doi: 10.52202/079017-1455. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/51173cf34c5faac9796a47dc2fdd3a71-Abstract-Conference.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. doi: 10.48550/arXiv.1803.05457. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024. doi: 10.1038/s41586-024-07421-0. URL https://www.nature.com/articles/s41586-024-07421-0.

Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/982572853408542621855 00e59bc0f28-Abstract-Conference.html. arXiv:2508.15260.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *ICML*, volume 202, pp. 10764–10799, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

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. In *NeurIPS*, volume 38, pp. 41340–41391, 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html. arXiv:2502.05171.

Jyotin Goel, Ipshita Bandyopadhyay, and Justin Shenk. The knowing-saying gap: When probes see errors that confidence misses. *arXiv preprint arXiv:2608.07528*, 2026. doi: 10.48550/arXiv.2608.07528. URL https://arxiv.org/abs/2608.07528.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021. URL https://openreview.net/pdf?id=d7KBjmI3GmQ.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *ICLR*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ. arXiv:2310.01798.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. doi: 10.48550/arXiv.2207.05221. URL https://arxiv.org/abs/2207.05221.

Jan Kirin. Operational proto-introspection in looped language models: Process-quality taps, executable branching, and the readout-control boundary. *arXiv preprint arXiv:2607.18553*, 2026. doi: 10.48550/arXiv.2607.18553. URL https://arxiv.org/abs/2607.18553v3.

Dharshan Kumaran, Viorica Patraucean, Simon Osindero, Petar Veličković, and Nathaniel Daw. How LLMs detect and correct their own errors: The role of internal confidence signals. *arXiv preprint arXiv:2604.22271*, 2026. doi: 10.48550/arXiv.2604.22271. URL https://arxiv.org/abs/2604.22271.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *ICLR*, 2024. doi: 10.48550/arXiv.2401.10480. URL https://arxiv.org/abs/2401.10480. arXiv:2401.10480.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *NeurIPS*, volume 36, pp. 46534–46594, 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/8c5f30296296d2ae402ebbd09aaa9c12-Abstract-Conference.html. arXiv:2410.21272.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *ICLR*, 2025. URL https://openreview.net/pdf?id=KRnsX5Em3W. arXiv:2410.02707.

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. doi: 10.48550/arXiv.2607.20519. URL https://arxiv.org/abs/2607.20519.

Philip Quirke and Fazl Barez. Understanding addition in transformers. In *ICLR*, 2024. URL https://openreview.net/pdf?id=rIx1YXVWZb.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *ICLR*, 2025. URL https://openreview.net/pdf/3b6624b631e0aa8b60e9db8890eda637165a01fa.pdf.

Kristian Schwethelm, Daniel Rueckert, and Georgios Kaissis. Depth-adaptive inference of looped language models via continuous depth batching. *arXiv preprint arXiv:2608.09444*, 2026. doi: 10.48550/arXiv.2608.09444. URL https://arxiv.org/abs/2608.09444.

Rituraj Sharma and Tu Vu. Dense supervision is not enough: The readout blind spot in looped language models. *arXiv preprint arXiv:2606.24898*, 2026. doi: 10.48550/arXiv.2606.24898. URL https://arxiv.org/abs/2606.24898.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. The limits of inference scaling through resampling. In *ICLR*, 2026. URL https://openreview.net/pdf/a64eb1288f0bc10366154b5a75b96b5fdef13bed.pdf. arXiv:2411.17501.

Yucheng Sun, Alessandro Stolfo, and Mrinmaya Sachan. Probing for arithmetic errors in language models. In *EMNLP*, pp. 8111–8128, 2025. doi: 10.18653/v1/2025.emnlp-main.411. URL https://aclanthology.org/2025.emnlp-main.411/. arXiv:2507.12379.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of ACL*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *NAACL*, pp. 4149–4158, 2019. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *EMNLP*, pp. 5433–5442, 2023. doi: 10.18653/v1/2023.emnlp-main.330. URL https://aclanthology.org/2023.emnlp-main.330/.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *ACL*, pp. 9426–9439, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023. URL https://openreview.net/pdf?id=1PL1NIMMrw.

Xiao-Wen Yang, Zi-Yu Han, Xi-Hua Zhang, Wen-Da Wei, Jie-Jing Shao, Lan-Zhe Guo, and Yu-Feng Li. Stabilizing recurrent dynamics for test-time scalable latent reasoning in looped language models. In *ICML*, 2026. URL https://icml.cc/virtual/2026/poster/61109. arXiv:2605.26733.

Aojie Yuan, Zhiyuan Julian Su, Haiyue Zhang, Yi Nian, and Yue Zhao. Hidden error awareness in chain-of-thought reasoning: The signal is diagnostic, not causal. *arXiv preprint arXiv:2605.09502*, 2026. doi: 10.48550/arXiv.2605.09502. URL https://arxiv.org/abs/2605.09502.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of ACL*, pp. 10495–10516, 2025. doi: 10.18653/v1/2025.findings-acl.547. URL https://aclanthology.org/2025.findings-acl.547/. arXiv:2501.07301.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *ACL*, pp. 1009–1024, 2025. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/. arXiv:2412.06559.

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. doi: 10.48550/arXiv.2510.25741. URL https://arxiv.org/abs/2510.25741.

A Reproducibility Details

Probe. Logistic regression (scikit-learn, C=1.0, `max_iter`=2000) on standardized features, with the standardization fit on the training fold only. All models run frozen on one 32GB GPU; for Qwen3-1.7B we probe all 29 hidden-state outputs (the embedding output and the 28 block outputs). Of the 3,578 completed 2.6B generations, 79 are shortened prefixes of gold with no wrong emitted digit; under the left-aligned token-level target every one of their digits is labelled correct. The features are the pre-normalization hidden state (dimension 2048 per loop step) at the position preceding each answer-digit token, taken from a single teacher-forced forward pass of the generated text (bf16 cast to fp32). Labels come from comparing the generated digits to gold, left-aligned; answers whose digit string cannot be aligned to the text span (for example after leading-zero collapse) are skipped. Evaluation uses GroupKFold with 5 folds grouped by problem; problem-level scores are the minimum over token scores. Detection CIs use 1000–5000 problem resamples; the updated selection and fixed-fraction retry CIs use 10,000 problem-clustered resamples (Appendix C).

Retry and cost protocol. Fixed-fraction temperature retry uses outer five-fold problem splits, training on greedy and sampled tokens from other problems. Within each evaluation fold the bottom 20% of greedy scores receive one sample at temperature 0.7 (the count is rounded within each fold). On 2.6B the length check applies to both greedy and the resample. The sample replaces greedy only if its score is strictly higher; a missing sample score is $-\infty$, retaining greedy even when both scores are $-\infty$. The 1.4B and adaptive-fraction retry analyses retain their original scorer without the length check, as marked in Table 10.

Nominal generation cost is K for K samples at depth 4, $1 + f$ for same-depth retry, and $1 + fu/4$ for depth-u retry. These conventions assume generations of comparable length and leave out offline data collection and scoring overhead. Teacher-forced rescoring and PRM scoring have additional overheads that are not measured by the nominal generation budget. Timing on 48 problems (three repeats, minimum elapsed time) gives depth-5 to depth-4 wall-clock ratios of 1.24 at batch 16 and 1.25 at batch 8, supporting the depth-cost convention for that run, though this is not an end-to-end latency measurement under adaptive batching.

Synthetic sets and their subsets. The dose–response run of §4 reads $0.785 \rightarrow 0.799 \rightarrow 0.951 \rightarrow 0.962$ on the 478-problem greedy+boost union. The *original set* was drawn with `random.Random(42)`, 50 problems per operand length 5–10: 300 generated, of which 258 completed. These 258 are the "258-problem greedy set" of the dose–response experiment, and the 253 of them that align under the original extraction script are the verifier set of Table 2. A 300-problem boost at 8–10 digits was added; greedy+boost gives 469 usable problems under the original extraction script (all results marked "original set") and 478 under the current script (the dose–response union). The *expansion set* uses seed 4242 with 583 problems per operand length: 3,498 generated, 389 truncated at 512 tokens and excluded, 3,109 usable. Of these, 498 differ from gold: 381 with the right number of digits and 117 with the wrong number, the latter being 65 shortened prefixes of gold with no wrong digit and 52 answers with a wrong digit, so 433 contain at least one wrong emitted digit. There is no operand-pair overlap with the original set. The *merged set* has 3,578 problems; 585 differ from gold, 506 with at least one wrong emitted digit and 79 shortened prefixes. All splits are by problem, identified by its operand pair. We checked programmatically (`audit_p0/p0_2_overlap.py`) that the fixed greedy bank added to every selection and retry training set (216 problems of length 8–10 from a separate generation run) shares no operand pair, unordered pair, or prompt with any evaluation problem; that no fold of any selection or retry run contains a training row from a test-fold problem; and that the merged set contains no duplicate operand pair, so grouping folds by problem is the same as grouping by operand pair. Probes are trained and evaluated on the merged set unless a result is marked "original set."

Prompts. Addition, with chain of thought: `"Question: What is {a} + {b}?\nAnswer step by step, then give the final numeric answer after '####'.\nAnswer:"` (`build_prompt` in `exp_arith.py` of the code release). GSM-hard uses the same zero-shot template as the base evaluation (`smoke_test.build_prompt`). Greedy collection uses `max_new_tokens`=512, or 1024 for Ouro-1.4B, which terminates with `\boxed{}`. Completion is checked before scoring, and truncated generations are excluded from probe data. For the 1.4B sampled candidate sets, 1024 tokens still leaves a sizeable max-length tail (13.9–19.4% of candidates; marker rate 0.869–0.891), so selection metrics use only problems whose greedy and

candidate hidden-state records align (886–888 of 900 for K=4; 888 for retry), with unavailable candidate members excluded from reranking. In single-resample retry a missing candidate score is set to $-\infty$, which keeps greedy; the attempted generation is still charged.

PRM baselines. Both PRMs read the same candidate texts as SLIP, truncated at the next `Question:` if the model ran on, and are segmented into steps by non-empty lines. This is the segmentation the PRMs' own recipes assume, but it does not line up with the addition columns: a column is written over a header line, bracketed display lines and a "write down, carry" line, so a candidate yields a few dozen PRM steps regardless of its operand length. Math-Shepherd follows its model card: each step is suffixed with the model's Cyrillic step tag and the step score is the softmax of the + against the − logit at the tag position. Qwen2.5-Math-PRM follows its card: the steps are joined by `<extra_0>` in an assistant turn under the model's own system prompt, and the step score is the positive class at each separator. For both, the candidate score is the minimum over step scores, the same bottleneck aggregation SLIP uses, and selection is the argmax over candidates. Neither PRM is adapted or fine-tuned to the column template, so this is a zero-shot out-of-domain use of them; the in-domain text baseline is the page–answer rule of Appendix B. Scripts: `exp_prm_baseline.py`, `exp_prm2.py`.

Text-repair baselines. Re-prompting and one-round Self-Refine are unconditional replacements on the 258 completed greedy problems, separate from the flagged re-decoding experiment. The saved re-prompting file contains 71 duplicate problem keys and one malformed line; valid records cover all 258 problems. Keeping the first or last valid record per problem gives the same outcomes: 224 correct after re-prompting and 221 after Self-Refine. Raw records are preserved; the audit script records the deduplication rule.

Candidate accounting for selection. The original majority and oracle calculations used all extracted answers, including last-number fallbacks from incomplete generations. SLIP requires an alignable answer span. On seed 1, $100/1{,}012$ extracted samples lack such a span; $90/100$ are wrong, versus $180/912$ alignable members. The 253 eligible problems have 4/3/2/1 alignable members on $176/58/15/4$ problems. Missingness is therefore not random. All current selection comparisons restrict every selector to the alignable members and then apply the same arithmetic length check. Sampling cost is charged for every attempted generation. Problems with at least one alignable sampled member remain eligible. If a selector has no candidate left after the length check, that problem remains in its denominator and counts as incorrect. Ties go to the first candidate in generation order, with greedy appended last. In seed 2 one problem has no alignable sampled member and is excluded (n=252); counting it as wrong changes SLIP4's three-seed mean from 0.956 to 0.955. Before the length check, restricting all extracted answers to alignable members raises majority from 0.884 to 0.892, Qwen PRM from 0.863 to 0.912, and the page–answer rule from 0.895 to 0.906; the any-of-4 ceiling falls from 0.974 to 0.972. Table 10 reports the current length-checked selection results; the next paragraph compares them with the original scorer. On expanded 1.4B, about 18% of samples are unalignable, mostly because generation hit its length limit. Its four-candidate SLIP accuracy remains below greedy after both filters. Canonical recomputation: `audit_p0/submission_selection.py`; earlier all-answer accounting: `audit_p0/same_pool_table2.py`. The twelve-feature difficulty baseline is `audit_p0/surface_baseline.py`. In the saved 1.4B depth sweep, the T=5 block at lengths 8–10 covers different problems. Non-monotonicity also holds on the 150 matched problems of lengths 5–7: $0.927/0.813/0.867/0.893$ at T=4/5/6/7, counting incomplete generations as wrong. The label-free T=4/3 analysis does not use the affected block.

The prefix blind spot and the length check. A strict prefix of gold has no emitted digit labelled wrong, so its score can approach 1. On the seed-1 2.6B verifier set, $11/912$ alignable samples are wrong prefixes; raw four-candidate selection picks one on $5/253$ problems (`audit_p0/prefix_blindspot.py`). Rejecting answers shorter than the longer positive operand removes 574 sampled members over three seeds and both models (48 on 2.6B, 526 on 1.4B), none correct. The added greedy candidate is also checked. On 2.6B, the check changes four-candidate selection from 0.938 to 0.956 (paired problem-clustered gain CI $[+0.9, +2.9]$ points), five-candidate selection from 0.945 to 0.967 $[+1.2, +3.6]$, and per-fold fixed-20% retry from 0.921 to 0.925 $[-0.1, +1.1]$. On 1.4B the corresponding selection means are $0.847 \rightarrow 0.856$ and $0.911 \rightarrow 0.917$. At K=16, the check rejects 67 sampled members and changes argmax from 0.913 to 0.988 $[+4.3, +11.1]$. It is only a partial remedy: with two d-digit operands, a d-digit prefix of a $(d{+}1)$-digit sum still passes (for example, $68812137 + 82909480$ with answer 15172161). The check applies to current synthetic selection and 2.6B fixed-fraction retry; detection, calibration,

distribution-matching diagnostics, adaptive retry, 1.4B retry, and the chained pipeline retain their stated original protocols.

The chained pipeline (§5.3). Each chain is $v_t = v_{t-1} + a_t$ for L=4 steps, with 120 chains drawn at seed 4242 (v_0 of five digits, addends of five to seven). Every step is a *separate* prompt in the addition template, so a resample regenerates only the failed step and the scalar it produces is what passes forward; nothing downstream is regenerated, which is why the cost stays near $1.2\times$. This experiment predates the length check and uses the raw probe score throughout. The gate is an absolute threshold, not a per-batch quantile: the classifier is fitted once on the synthetic pool (merged greedy plus the T=0.7 candidate tokens) and τ is then set to the 20th percentile of the per-problem minimum score over the greedy part of that pool alone ($\tau = 0.979$); both are frozen thereafter. A step is flagged when its minimum digit score falls below τ, or when its answer cannot be parsed or aligned, which scores 0. The flagged step is redrawn once at temperature 0.7 and the higher-scoring of the two is committed. The random control draws at the probe's realized flag rate, though on the fixed control seed it fires somewhat more often than the probe (0.26 against 0.22 of steps), so it is if anything the better-funded control; the oracle flags exactly the wrong steps. No chain step shares an operand pair with the pool the probe was fitted on (`audit_p0/chain_overlap.py`). Conditions share the chain bank, so intervals are chain-clustered paired bootstraps over three retry seeds (`audit_p0/chain_ci.py`); greedy is deterministic and runs once. Accuracies are 0.442 [0.358, 0.533] with no gate, 0.583 [0.506, 0.658] with the probe gate, 0.461 [0.378, 0.542] random and 0.642 [0.569, 0.711] oracle, at 1.00/1.22/1.26/1.17× steps generated.

B Additional Probe Controls and Sample Composition

Table 4 lists the controls referred to in §4, all on the merged 3,578-problem 2.6B set with the GroupKFold protocol of the paper, and Table 5 breaks that set down by operand length.

Patching endpoints and controls. Each of 345 distinct problems supplies one wrong target digit and a correct-run donor. At the tested late-loop patch, all 68 output changes correct that digit. A wrong-run donor is also available on 158 problems: correct-run donors correct 29 digits, whereas wrong-run donors change 36 and correct 19. The latter donors have a gold digit at the patched location on 46 problems despite an incorrect full answer. In a post-hoc split, the 112 locally wrong donors yield 17 changes to another wrong digit and no corrections, versus 20 corrections from correct-run donors on those same problems. All 36 wrong-run-donor changes match the donor digit. These are local next-digit endpoints under teacher forcing, not measurements of complete-answer repair, and donor selection uses gold labels. The result supports partial donor-digit transfer but does not establish an autonomous repair procedure. Probe-direction ablation remains separate: it changes no tested output digits on 498 slip and 100 correct problems.

Error types. Counting all 585 answers that differ from gold (a wider denominator than the same-length census in §4), the types are: digit substitutions other than ±1, 252 (43%); spurious carry, 155 (27%); shortened prefix of gold with no wrong digit, 79 (13.5%); wrong length, 67 (11.5%); dropped carry, 22 (4%); mixed, 10. Token AUROC by type ranges over 0.974–0.994 for every type that contains a wrong emitted digit; for the prefix-only type it is undefined. Operand carry counts span 0–9, and the error rate rises from 8% at one carry to 37% at seven; 437 distinct (length, carry-pattern) configurations occur among the erroneous problems. The prospective control of §4 is `audit_p0/prospective_difficulty.py`: the twelve problem-observable features reach 0.757 token and 0.834 problem AUROC on the pre-error positions, difficulty-matched pairing over 39 (length, carry-count) cells leaves the hidden probe at 0.983 [0.976, 0.990] against 0.984 unmatched, and a model with both feature sets reads 0.983, the same as the hidden state alone. The features are stronger inside a stratum than pooled, reaching 0.93 at nine- and ten-digit operands against 0.834 overall, so the pooled figure understates how well difficulty alone predicts this target; the probe leads it in every stratum regardless. Two features, operand length and carry count, give only 0.628, so most of what the baseline has is finer per-column structure. Only 14 of the 506 erroneous answers have a wrong most significant digit. At the operating point of the retry policy the probe flags 14/14 of them, against 487/492 of the rest, and its top-1 pick is a wrong digit in 6/6 of the ones that are not wrong in every digit; the counts are small enough that these rates are indistinguishable (`audit_p0/msd_breakdown.py`).

The page–answer rule. The rule in §5 parses every column line of the chain of thought ("a+b(+1) = d", "write down w", "carry k") and flags an answer whose digits differ from the digits it reassembles from those lines. It parses all $3{,}578/3{,}578$ generations. The candidate selector also checks the written arithmetic and carries for consistency before taking a majority vote among the unflagged candidates. Of the 499 wrong-digit answers that reassembly catches, the written columns reassemble to gold in 363 and to neither gold nor the answer in 136. In the remaining 7 slips the written columns reassemble to the wrong answer itself, and the rule misses them.

Calibration. Before the arithmetic length check, on the K=16 candidate collection (253 eligible problems), token ECE is 0.023 and the Brier score 0.027; ordering tokens by score gives an AURC of 0.005 against a token error rate of 0.066 unordered. These are token-level numbers on that collection, not merged-set or problem-level metrics. Aggregating to candidate scores raises ECE to 0.053, and after argmax selection the problem-level ECE is 0.091. Token calibration does not transfer to selection. For comparison, a gold-trained TF-IDF classifier over the generated text reaches 0.737 problem AUROC on the merged slip target, below both the reassembly rule (0.884) and SLIP (0.991).

Table 3: Label-free training (§5.1). Pseudo-labels mark a digit suspect when the two recomputations disagree; the probe is trained on pseudo-labels only and evaluated out of fold against gold. Pseudo-label precision and recall are reported in the detection direction, treating erroneous digits as positives, unlike the probe target itself. 2.6B: 253-problem verifier set, T=4 vs. 5; 1.4B: original 292-problem set, T=4 vs. 3; Huginn: 272-problem few-shot set, r=8/16/32, 10-fold in the saved run; GSM-hard: digit agreement across eight alternative generations. †The GSM-hard supervised value is the exact-answer recomputation of Appendix D, evaluated under a different protocol from the label-free run, so the two are not a paired comparison.

Model	Pseudo-label P / R / F1	Label-free (token / problem)	Supervised (token / problem)
Ouro-2.6B	0.58 / 0.69 / 0.63	0.893 [0.828, 0.941] / 0.834	0.977 / 0.958
Ouro-1.4B	0.64 / 0.56 / 0.59	0.858 [0.791, 0.917] / 0.889	0.914 / 0.957
Huginn-0125	—	0.721 / —	0.839 / —
GSM-hard (2.6B, problem)†	—	0.773	0.830

Table 4: Probe controls on the merged 3,578-problem 2.6B set (§4): token AUROC with 95% bootstrap CIs, problem AUROC, and top-1 localization accuracy. "Prospective" rows use gold-defined pre-error positions in wrong answers and all digit positions in no-slip answers; first-digit errors are excluded.

Probe / labeling	token AUROC	problem	top-1
Logistic, step 3 (main text)	0.985 [0.979, 0.990]	0.991	0.972
first wrong digit only (later ones dropped)	0.987 [0.981, 0.993]	0.974	0.968
first wrong digit only (later ones relabeled correct)	0.982 [0.974, 0.989]	0.971	0.907
PCA-64 + logistic	0.971 [0.963, 0.977]	0.953	0.924
MLP, 2 layers (256)	0.985 [0.980, 0.990]	0.989	0.968
MLP, 3 layers (256, 256)	0.981 [0.969, 0.988]	0.990	0.968
Trajectory logistic, steps 0–3 concatenated	0.990 [0.985, 0.993]	0.991	0.976
Prospective: positions before first wrong digit only	0.954 [0.946, 0.961]	0.983 (mean-pool)	—
main-text probe applied prospectively	0.588	—	—
12 problem-observable features, same positions	0.757 [0.743, 0.770]	0.834	—

C Statistical Protocol

Unless stated otherwise, confidence intervals are percentile bootstraps over problems, and localization proportions use Wilson intervals. For updated selection and 2.6B fixed-fraction retry, we resample problem IDs with their outcomes across all seeds kept together (10,000 resamples, seed 123), then average the per-seed eligible accuracies with equal weight. A three-seed mean is the arithmetic mean of the per-seed accuracies, each on its own eligible denominator; a CI quoted for one seed is

Table 5: Merged 2.6B set by operand length d: problems, erroneous problems (with a wrong emitted digit), error rate, and detection (token AUROC with 95% CI, problem AUROC, and top-1 localization). The erroneous column sums to 506 and the top-1 denominators to 498: the 8 answers in which every emitted digit is wrong are excluded from top-1, as in §4.

d	problems	erroneous	err. rate	token AUROC	problem	top-1
5	633	42	6.6%	0.969 [0.922, 0.998]	0.981	40/41
6	633	59	9.3%	0.984 [0.969, 0.996]	0.990	56/57
7	631	72	11.4%	0.988 [0.980, 0.994]	0.990	69/70
8	644	131	20.3%	0.992 [0.989, 0.996]	0.994	129/131
9	572	98	17.1%	0.989 [0.979, 0.995]	0.996	96/97
10	465	104	22.4%	0.976 [0.961, 0.988]	0.988	94/102

not a CI for the mean. The bootstrap resamples evaluated problems given the fitted out-of-fold scores and does not retrain the whole procedure in each replicate. The settings we explored, including the aggregation rules, were not fixed in advance, and we make no claim of a multiplicity-adjusted test family.

Adaptive retry and exploratory aggregation have additional qualifications.

Historical depth retry. The full nested evaluation code for an earlier depth-retry analysis is not preserved. We exclude its performance estimate from the result tables and from the evidence supporting our conclusions.

Inner cross-validation for the temperature retry. The outer evaluation scores are clean: the scored problem is never in the probe's training data. An earlier version of the inner routine that chose the adaptive retry fraction excluded only the inner validation problems from the greedy pool, so the greedy tokens and labels of the outer test problems could influence the chosen fraction. We corrected the inner routine to exclude both the outer test and the inner validation problems and reran the analysis from the saved hidden states (`audit_p0/retry_seed_sim_fixed.py`). On 2.6B the corrected adaptive-fraction accuracies are unchanged (0.921 / 0.909 / 0.917, mean 0.916): the chosen fraction moved in two of fifteen folds, to values that give the same test-fold outcome. On 1.4B the corrected mean is 0.884 (0.890 / 0.883 / 0.881), up from 0.882, so the leak had not inflated the result. Fixed-20% retry does not use the inner step. The current 2.6B result additionally applies the length check and selects its budget within each outer fold; the adaptive results above do not include that check. We chose 20% after looking at the sweeps. The 1.4B logistic fits vary in the third decimal with BLAS thread count. Updated candidate selection and 2.6B fixed-fraction retry use single-thread BLAS; the separately marked 1.4B retry values retain their recorded evaluation.

BBH and the aggregator. We recorded the census-based predictions before running the BBH probe and self-consistency experiments, but did not fix the aggregation rule in advance. Weighted voting, both at K=16 and on BBH, was tried after argmax failed. At K=16 that failure turned out to be the prefix blind spot of Appendix A, and with the length check argmax no longer needs the weighted vote; we report argmax there and keep both on BBH, where the same explanation does not apply. The weighted vote sums each candidate's probe score by answer string and includes the greedy answer as a fifth voter; no tally was tied. The original BBH argmax script broke exact score ties (probe probabilities saturated at 1.0) in favour of the correct candidate. Re-running the aggregation with index-based and random tie-breaking (`audit_p0/bbh_tiebreak_audit.py`) finds one affected problem across the three object-counting seeds, moves that seed's argmax by 0.4pt, and leaves the recorded weighted-vote mean unchanged (0.861; +4.1pt over greedy, problem-clustered CI [+0.7, +7.5]). The earlier +2.8pt comparison used four sampled voters for majority but five for weighted voting. On identical alignable candidates including greedy, majority reaches $0.868/0.860/0.856$ (mean 0.861), matching the recorded weighted-vote mean. A single-thread refit gives weighted voting 0.860 versus majority 0.861 (paired difference -0.13pt, CI [-2.27, +2.00]), so neither evaluation supports superiority over matched voting. The candidate-discrimination AUROCs pool candidates across problems rather than ranking within a problem, which flatters them, so we measured the within-problem version (`audit_p0/bbh_within_problem.py`). On object counting the pooled 0.81 becomes a pairwise within-problem ranking accuracy of $0.61/0.61/0.65$ over the three seeds, each with a bootstrap lower bound above chance, and choosing the top-scored candidate beats drawing one at random by

$3.0/2.9/3.8$ points, an interval that excludes zero in one seed of three. Between 101 and 120 of the 250 problems have both a right and a wrong candidate; on the rest no selector can change the outcome. On multistep arithmetic only 23 problems are rankable, so its within-problem value of 0.55 $[0.38, 0.72]$ shows an absence of power rather than an absence of signal, and the gap between the two tasks is not established at the within-problem level.

D Detection Baselines in the Wild (Full Comparison)

Table 6 compares signals for predicting whether the greedy GSM-hard answer is wrong. The exact-answer recomputation gives SLIP an AUROC of 0.830 on 676 eligible problems. That beats the answer-logprob baseline, and the probe also supplies token-level scores, although for triggering a full recomputation a sequence-level score would do. The detection collections contain 132 and 676 eligible problems; five-fold training uses about four fifths of each per fold. Unlike every baseline in the table, the probe needs a one-time offline training set from the task distribution, labeled either with gold or, as in §5.1, with disagreement, and a separate set per task (§6); the "Labels" column makes this cost explicit. All DeepConf numbers in the paper are a small-K offline reduction of DeepConf (Fu et al., 2026), whose headline experiments use K=512; the collapse of its filter at K=4 is partly a consequence of that reduction.

Table 6: Wrong-answer detection on GSM-hard, Ouro-2.6B: AUROC for detecting errors in greedy answers, with percentile bootstrap CIs over problems. Each signal is evaluated on the largest GSM-hard subset available to it (n=200–1,000), so rows are not paired. The SLIP row is the saved exact-answer recomputation on 676 eligible problems; an earlier protocol, which we cannot reconstruct exactly, gave 0.842; its CI is not carried over to 0.830. "Extra cost" is generation beyond producing the answer once, relative to one greedy pass; "Labels" is the number of labeled problems a signal needs offline. Bold: best AUROC.

Signal	Extra cost	Labels	AUROC	Localizes?
Answer-token logprob	0	0	0.64 [0.60, 0.68]	no
Verbalized conf. (Tian et al., 2023)	1 pass	0	0.52	no
p(True) (Kadavath et al., 2022)	1 pass	0	0.65	no
SLIP (ours; exact-answer target)	TF pass	~700	0.830	yes
SC answer-entropy, K=6	5×	0	0.838	no
SC agreement, K=6	5×	0	**0.886**	no

On clean synthetic addition, p(True) matches the probe at the problem level (0.987 vs. 0.982 on the original set, overlapping CIs). The probe's advantages are the in-the-wild gap above and digit-level localization, which a sequence-level score cannot provide. Table 7 collects the failure modes the probe does not catch.

Table 7: Failure modes the probe does not catch, with the measured detection level and the section where each is reported.

Failure mode	Measured	Where
Missing answer suffix with no wrong emitted digit	outside the digit-label target	§4 census; Appendix B
Transcription errors (reversal/tracking tasks)	0.64 substitution-only	§6
Reasoning failures (GSM8K residual errors)	0.645 detection	§5.2
Zero-shot cross-task weight transfer	0.62–0.69	§6
Recovering the *true* digit (vs. flagging)	0.159 (chance 0.1)	§5
Huginn knowledge tasks (cross-architecture)	0.51–0.61, flat	§6

E Generality Results

Table 8 lists the numbers summarized in words in §6.

Table 8: Results behind §6. AUROC values are problem-level unless marked token-level; intervals are paired 95% bootstrap CIs. Multiple-choice: 1,392 problems. BBH and the pipeline: three seeds; the pipeline uses 120 four-step chains. Huginn: 272-problem 4–5-digit few-shot set.

Experiment	Quantity	Value
Other tasks	in-task probe AUROC: addition (token) / median / counting / GSM-hard	0.985 / 0.792 / 0.715 / 0.830
	addition-trained probe on the other tasks (token)	0.62–0.69
Knowledge	answer log-probability AUROC / probe / probe across loop steps	0.825 / 0.754 / 0.616 → 0.754
BBH	candidate discrimination, pooled / within problem: object counting	0.81 / 0.62
	multistep arithmetic	0.53 / 0.55
	problems with both a right and a wrong candidate	101–120 / 23
	object counting weighted vote / majority, same five candidates; gain over greedy	0.861 / 0.861; +4.1pt [+0.7, +7.5]
Pipeline	no gate / probe gate / random gate / oracle gate	0.442 / 0.583 / 0.461 / 0.642
Transcription	reversal and tracking errors, detection AUROC	0.64
Huginn	detection gain r=1 → 16 (paired)	+0.037 [+0.003, +0.075]
	recurrence candidates (r=8/16/32 greedy, read at r=16): best single / any-of-3 / probe-weighted vote gain	0.520 / 0.718 / +7.9pt [+1.5, +13.9]

F Full Cost–Accuracy Landscape

Table 9 lists every strategy in Table 2 together with the dominated settings we left out of the main text. Table 10 gives the per-seed numbers behind the three-seed means, and Table 11 the distribution-matching diagnostics.

Table 9: Full cost–accuracy landscape on multi-digit addition, Ouro-2.6B. Three-seed selection uses 253/252/253 eligible problems and retry uses 253. Costs are nominal generation budgets. Top: baselines at their native budgets. Bottom: the budget curve of SLIP, with the same probe family fit separately to each candidate collection; all candidate selectors use the same length check, including 16-candidate argmax (§5). Rows marked ‡ are on the 258-problem greedy set, as in Table 2; its greedy accuracy is $225/258 = 0.872$, against $220/253 = 0.870$ on the verifier set. Uniform depth 5 counts incomplete answers as failures ($210/253 = 0.830$). Bold: best in column, ceilings excluded. Majority ties, possible at small K, go to the first sample. Dominated settings not shown: SC K=2 (0.683) and SC K=4 at temperature 1.0 (0.802).

Strategy	Cost	Accuracy
Greedy (depth 4)	1.0×	0.870
Uniform deeper (all at depth 5)	1.25×	0.830
Re-prompting‡	2×	0.868
ESC (w=2)‡	2.9×	0.860
Self-Refine (1 round)‡	3×	0.857
SC majority, K=8‡ (three seeds: 0.934–0.953)	8×	0.947
SC majority, K=16, same filtered candidates	16×	0.984
SLIP (ours)		
2 candidates (greedy + 1 resample, flagged 20%; three-seed mean)	1.2×	0.925
4 candidates (SC samples; seed 1 / mean)	4.0×	0.953 / 0.956
majority over the same 4 / 4+greedy (means)	4.0× / 5.0×	0.897 / 0.937
5 candidates (SC + greedy; seed 1 / mean)	5.0×	0.972 / 0.967
16 candidates (argmax)	16.0×	**0.988** [0.972, 1.000]
any-of-5 (three-seed mean, same candidates) / any-of-16 ceilings	—	0.980 / 1.000

Table 10: Per-seed results behind §5. Current selection uses alignable members and the same length check for every selector, including greedy as a fifth candidate. Ouro-2.6B: $n = 253/252/253$ for selection and 253 for retry; Ouro-1.4B: $n = 888/886/887$ for selection and 888 for retry. Rows marked "no check" retain the earlier scorer. Adaptive retry uses corrected nested CV (Appendix C).

Model	Strategy	Seed 1 / 2 / 3 (mean)
2.6B	Greedy	0.870 / 0.869 / 0.870 (mean 0.869)
2.6B	Majority, 4 candidates	0.885 / 0.905 / 0.901 (mean 0.897)
2.6B	SLIP, 4 candidates	0.953 / 0.956 / 0.960 (mean 0.956)
2.6B	SLIP, 5 candidates	0.972 / 0.960 / 0.968 (mean 0.967)
2.6B	Majority, 5 candidates	0.925 / 0.952 / 0.933 (mean 0.937)
2.6B	Any-of-4 ceiling	0.964 / 0.976 / 0.976 (mean 0.972)
2.6B	Any-of-5 ceiling	0.976 / 0.980 / 0.984 (mean 0.980)
2.6B	Math-Shepherd PRM	0.834 / 0.841 / 0.881 (mean 0.852)
2.6B	Qwen2.5-Math-PRM-7B	0.901 / 0.909 / 0.941 (mean 0.917)
2.6B	Page–answer rule, 4 candidates	0.909 / 0.925 / 0.905 (mean 0.913)
2.6B	Page–answer rule, 5 candidates	0.917 / 0.940 / 0.917 (mean 0.925)
2.6B	Temperature retry, fixed 20%	0.929 / 0.925 / 0.921 (mean 0.925)
2.6B	Adaptive retry, no check	0.921 / 0.909 / 0.917 (mean 0.916)
1.4B	Greedy	0.865 / 0.865 / 0.865 (mean 0.865)
1.4B	Majority, 4 candidates	0.789 / 0.791 / 0.802 (mean 0.794)
1.4B	SLIP, 4 candidates	0.863 / 0.857 / 0.849 (mean 0.856)
1.4B	SLIP, 5 candidates	0.920 / 0.921 / 0.910 (mean 0.917)
1.4B	Majority, 5 candidates	0.864 / 0.860 / 0.869 (mean 0.864)
1.4B	Any-of-5 ceiling	0.955 / 0.958 / 0.957 (mean 0.957)
1.4B	Fixed-20% retry, no check	mean 0.886
1.4B	Adaptive retry, no check	0.890 / 0.883 / 0.881 (mean 0.884)

Table 11: Distribution-matched training (§5). The discrimination rows are diagnostics from the original analysis: greedy rows report token AUROC, and the sampled-candidate AUROCs pool candidates across problems rather than ranking within a problem. The selection rows measure within-problem choices directly. "Greedy-only" probes are trained on greedy generations; "mixed" probes add temperature-sampled generations. The selection diagnostics here do not include the later length check; 1.4B also uses its original set, so these differ from the current standardized selection numbers. The equal-token control holds the training set at about 7.2k tokens (1.4B; mean ± sd over folds).

	Ouro-1.4B	Ouro-2.6B
Discrimination on greedy candidates (in-distribution)	0.914	0.966
Discrimination on sampled candidates, greedy-only probe	0.691	0.895
Discrimination on sampled candidates, mixed probe	0.872	0.969
4-candidate selection, greedy-only probe vs. majority	0.736 vs. 0.760	0.930 vs. 0.892
5-candidate selection, greedy-only → mixed	0.822 → 0.904	0.953 → 0.964
Equal training tokens, greedy-only → mixed	0.754 ± 0.036 → 0.869 ± 0.023	0.924 ± 0.031 → 0.933 ± 0.013