LET IT SETTLE: ALLOCATING TEST-TIME COMPUTE IN LOOPED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Inference compute in looped language models can be allocated to additional loops or longer chains of thought. Under a fixed budget, this allocation involves a trade-off because each additional loop increases both prompt-processing and per-token generation costs. We address this with SETTLE, an allocator based on answer settling, which separates when intermediate answers stop changing from whether they are correct. Using these measurements and a small labelled calibration set, SETTLE selects a loop count and chain-of-thought token limit under an average compute budget. We evaluate six models from the Ouro, Huginn, and Recurrent-Llama families on ten reasoning benchmarks, bound selection loss in terms of accuracy-estimation error, and isolate the contributions of estimation, budget sharing, and validation. We find that the best depth and chain length vary by model, task, and budget, and that most of the improvement comes from task-level allocation, with smaller benefits from varying settings across prompts. At half the baseline cost, SETTLE-H is 9.8 points more accurate than token-only allocation where both are feasible. Including prompt processing, it reaches baseline accuracy at a median of 68% of baseline cost, allowing a one-point tolerance in four cases. Budget-conditioned self-distillation further improves GSM8K accuracy by 36 points at a 32-token budget.

1 INTRODUCTION

Looped transformers refine token representations by repeatedly applying shared layers, increasing effective depth without adding parameters and showing promising reasoning performance (Saunshi et al., 2025; Geiping et al., 2025; Zhu et al., 2025). Recent models, including Ouro, Huginn, and Recurrent-Llama, let inference choose the loop count (Zhu et al., 2025; Geiping et al., 2025; McLeish et al., 2025). Combined with chain-of-thought generation, they expose two compute axes: recurrent depth and reasoning length. Prior work largely studies them separately: depth studies add loops (Geiping et al., 2025); elastic-depth training and adaptive exits vary compute across budgets or tokens (Jeddi et al., 2026; Bae et al., 2025; Popescu et al., 2026); and budgeted reasoning varies chain length at fixed depth (Muennighoff et al., 2025; Aggarwal & Welleck, 2025; Han et al., 2024; Huang et al., 2025). Figure 1 shows that longer chains do not always recover the accuracy lost from fewer loops, while at higher loop counts much shorter chains can approach full-chain accuracy. Loop count and chain length are therefore not interchangeable. We ask: for a given model and task under a fixed compute budget, how should they be chosen to maximize accuracy?

We study six looped models from three families on ten reasoning benchmarks, evaluating loop-count and chain-length combinations and measuring prompt-processing and generation costs. We define *answer settling* as the earliest measured token limit after which the answer stops changing. Settling needs no correctness labels but does not imply correctness, so a small labeled calibration set estimates the accuracy of settled and unsettled answers separately. We then formulate **SETTLE** (**S**tability-**E**stimated **T**runcation of **T**hought and **L**oop **E**xecution), which uses these estimates to choose a loop count and chain-of-thought token limit under an average compute budget. Calibration determines the rule from each setting's estimated accuracy and cost; new prompts receive settings before generation without trial runs. We also study budget-conditioned self-distillation, which teaches the model to answer within a stated token budget. Our contributions are:

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.

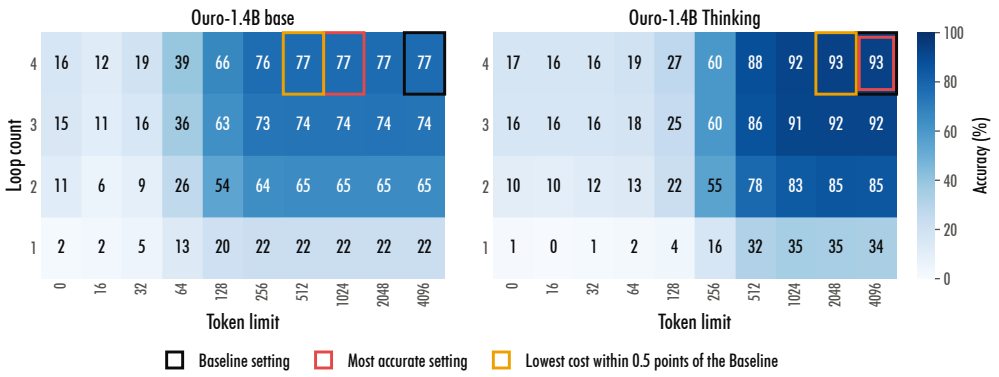

Figure 1: Accuracy (%) of Ouro-1.4B base (left) and Thinking (right) on GSM8K across loop counts and token limits. Boxes mark the baseline (ink), highest accuracy (red), and lowest-cost setting within 0.5 points of baseline accuracy (yellow)

- We characterize the joint effects of recurrent depth and chain length across six looped models and ten benchmarks. We identify answer settling, the point after which the answer read from a chain no longer changes, as a measurable event in looped models and decompose accuracy into the share of settled questions and the correctness of their settled answers, thereby separating stabilization speed from correctness.
- We introduce SETTLE-H, which selects loop count and token limit under a fixed average compute budget. At half the baseline cost, it outperforms Token-only by 9.8 points where Token-only is feasible and reaches baseline accuracy at a median of 68% of the baseline compute.
- We show most gains come from task-specific loop-count and token-limit choices; prompt-specific choices add less than one point at half the baseline cost. Settling estimates improve selection with 10 to 30 labeled questions and match direct accuracy lookup from about 50.
- We train a looped model to answer within a budget stated in the prompt by self-distilling. This improves GSM8K by 36.1 points over the base model at a 32-token budget, adds to SETTLE, and moves the settle point earlier through changes in the first six layers, although some tasks lose accuracy at longer budgets.

2 PRELIMINARIES

Settings. We consider a looped model with a shared block of L layers and L_{fixed} further layers. For each token, it applies the shared block k times and the further layers once, where the inference-time loop count k ranges up to $K_{\max}$, the largest measured value. A token limit T caps the chain of thought. The setting $u = (k, T)$ has task accuracy $A_k(T)$. We keep the weights fixed except in Section 4; Table 1 lists the symbols.

Scored answer. At each loop count, we generate one chain per question until the model stops and extract answers at ten token limits: 0 and the powers of two from 16 to 4096. At T, we score the model's final answer if it appears within the first T tokens; otherwise, we score the answer to a request appended after the cut. This request does not feed back into the chain, so one generation per loop count serves all ten limits.

Cost. A block pass applies one layer to one token, and the cost of a token at k loops is $L_{\text{fixed}} + kL$ block passes. Prompt tokens are charged at the same rate. The expected cost of the setting (k, T) on prompt i is

$$\widehat{C}_i(k, T) = (L_{\text{fixed}} + kL)\left[Q_i + \bar{\ell}_k(T) + R_i\right], \qquad \bar{\ell}_k(T) = \mathbb{E}_{\text{cal}}\left[\min\{\ell_k, T\}\right], \tag{1}$$

where ℓ_k is the length at which the model stops at k loops, R_i is the answer reserve, and Q_i is the number of prompt tokens charged. Unless stated otherwise, we use the prefix-cached cost scheme

Table 1: Notation.

Symbol	Meaning	Symbol	Meaning
k, $K_{\max}$	loops per token, largest measured	T	token limit on the chain
$u = (k, T)$	a setting	$A_k(T)$, $\widehat{A}$	accuracy of a setting, its estimate
$\widehat{C}_i(k, T)$	cost of a setting for prompt i	X, λ	budget fraction, cost multiplier
Q_i, R_i	prompt tokens charged, answer reserve	$\bar{\ell}_k(T)$	expected chain length under T
$J_{i,k}$	settle time of question i at k loops	$G_k(T)$, $p_k(j)$	settled share by T, share settling at j
$c_k(j)$	correctness of the group settling at j	$l_k(T)$	correctness of the unsettled at T

and assume that a serving system computes the few-shot block S_i of a task once. Thus, a prompt of P_i tokens is charged $Q_i = P_i - S_i$.

Budget. The baseline setting runs $K_{\max}$ loops until the model stops, and the baseline cost is defined to be its mean cost. A budget X is a fraction of the baseline cost. We specify the loop counts, the answer extraction, two other cost measures and the comparison policies in Appendix B.

3 Answer Settling

The answer extracted from a chain of thought can change as the chain grows longer. We study *answer settling*, the point after which the extracted answer stays the same at every longer measured token limit. Note that settling is found by comparing the model's answers with each other, so it requires no correctness labels.

3.1 Settle Time and Settled Share

Fix a loop count k, and let $T_0 < \cdots < T_{\max}$ be the measured token limits. Let $r_{i,k}(T)$ be the scored answer to question i at limit T (Section 2), and $Y_{i,k}(T) \in \{0, 1\}$, its correctness. Probabilities and expectations are over the questions of a task.

Definition 1 (Settle time)**.** *The settle time $J_{i,k}$ is the smallest measured token limit J such that $r_{i,k}(T) = r_{i,k}(T_{\max})$ at every measured token limit $T \geq J$. If the answer at $T_{\max}$ does not parse, $J_{i,k} = \infty$. The settled share is $G_k(T) = \mathbb{P}(J_{i,k} \leq T)$, and the share settling at token limit j is $p_k(j) = \mathbb{P}(J_{i,k} = j)$.*

Thus, $G_k(T)$ is the fraction of questions whose answer at T already equals the answer at every longer measured limit. Section 5.5 tests whether settling is visible in the hidden states of Ouro-1.4B.

3.2 Accuracy Decomposition

A settled answer can still be wrong, so we measure correctness separately for settled and unsettled questions. Let $A_k(T) = \mathbb{E}[Y_{i,k}(T)]$ be the accuracy. The settled correctness $c_k(j) = \mathbb{E}[Y_{i,k}(j) \mid J_{i,k} = j]$ is the accuracy on the questions that settle at j, and the unsettled correctness $l_k(T) = \mathbb{E}[Y_{i,k}(T) \mid J_{i,k} > T]$ is the accuracy on the questions not yet settled at T. A conditional mean over an empty group receives zero weight.

Lemma 1 (Accuracy by settling status)**.** *For every measured token limit, $A_k(T) = G_k(T)\,\bar{c}_k(T) + [1 - G_k(T)]\,l_k(T)$, where $\bar{c}_k(T) = \mathbb{E}[Y_{i,k}(T) \mid J_{i,k} \leq T]$ is the accuracy on the questions settled by T.*

Lemma 2 (Accuracy by settle time)**.** *Suppose correctness is a function of the scored answer. Then for every measured token limit,*

$$A_k(T) = \sum_{j \leq T} p_k(j)\, c_k(j) + \big[1 - G_k(T)\big]\, l_k(T). \tag{2}$$

Lemma 2 splits the settled term by settle time. A question that settles at j has the same scored answer at every measured limit $T \geq j$, so its correctness does not change. Appendix A gives the proofs and discusses the numerical gap caused by answer normalisation.

3.3 Effect of a Change of Setting

A change of loop count or token limit can change both the settled share and the correctness of settled answers. Let $S_k(T) = \sum_{j \le T} p_k(j)\, c_k(j)$ and $U_k(T) = [1 - G_k(T)]\, l_k(T)$ be the settled and unsettled terms of Eq. (2). Under the assumption of Lemma 2, $\bar{c}_k(T) = S_k(T)/G_k(T)$ whenever $G_k(T) > 0$.

Proposition 1 (Change of setting). *For two settings $u = (k, T)$ and $u' = (k', T')$ with $G_k(T) > 0$,*

$$A_{k'}(T') - A_k(T) = \underbrace{\big[G_{k'}(T') - G_k(T)\big]\,\bar{c}_k(T)}_{\textit{settling speed}} + \underbrace{G_{k'}(T')\,\big[\bar{c}_{k'}(T') - \bar{c}_k(T)\big]}_{\textit{settled correctness}} + \underbrace{U_{k'}(T') - U_k(T)}_{\textit{unsettled answers}}. \tag{3}$$

The second term is taken as zero when $G_{k'}(T') = 0$.

The first term is the change in settled share, at the old settled accuracy. The second is the change in settled accuracy, weighted by the new settled share. The third is the change in the unsettled term.

3.4 Estimation From Few Labels

We estimate the settling distributions $\widehat{G}_k$ and $\hat{p}_k$ from all calibration answers, without labels, and the correctness rates from a labelled subset. This gives two accuracy estimators:

$$\widehat{A}_{\mathrm{pool}}(k, T) = \widehat{G}_k(T)\,\hat{c}_k + \big[1 - \widehat{G}_k(T)\big]\hat{l}_k, \tag{4}$$

$$\widehat{A}_{\mathrm{res}}(k, T) = \sum_{j \le T} \hat{p}_k(j)\,\hat{c}_k(j) + \big[1 - \widehat{G}_k(T)\big]\hat{l}_k(T). \tag{5}$$

The pooled estimator uses one settled rate and one unsettled rate per loop count, shared across token limits. The resolved estimator estimates the settled rate per settle time and the unsettled rate per token limit. Both take the group proportions from all calibration answers, so labels are needed only for the correctness rates. If every setting within the budget is estimated to within ε, the setting with the highest estimate has accuracy within 2ε of the best feasible setting (Proposition 2, Appendix A).

4 Allocation

For each model, task and budget X, SETTLE fits a policy on calibration questions that uses the settling estimate of Section 3.4 to assign each prompt a loop count and chain-of-thought token limit before generation, with cross-validation against a simple reference. One chain per question and loop count provides answers at all measured token limits, yielding the settling distribution; a labelled subset provides correctness rates. Each setting receives a task-wide score $\widehat{A}_{\mathrm{res}}(k, T)$ from Eq. (5) and a prompt-specific cost $\widehat{C}_i(k, T)$ from Eq. (1) that rises with prompt length. An average budget transfers savings between prompts, whereas a per-prompt budget strands compute on short prompts. The policy is

$$\pi_\lambda(i) \in \arg\max_{(k,T) \in \mathcal{F}} \big\{\widehat{A}_{\mathrm{res}}(k, T) - \lambda\, \widehat{C}_i(k, T)\big\}, \tag{6}$$

where $\mathcal{F}$ is a candidate family of settings and $\lambda \ge 0$ trades estimated accuracy against cost. Because mean estimated cost is nonincreasing in λ, bisection selects the smallest λ meeting the budget on calibration prompts. The budget is infeasible if the cheapest setting in $\mathcal{F}$ for every prompt still exceeds it on average.

Because the score in Eq. (5) is estimated, a fitted policy may underperform the reference, which uses the largest loop count and longest token limit that fit the budget (Appendix E.4). We therefore require improvement beyond the calibration set. Each of four candidate families is fitted on 70% of the calibration questions and evaluated by its accuracy margin over the reference on the other 30%, then the subsets are swapped. We retain a family only if both margins exceed 0.5 bootstrap standard deviations (2000 resamples). SETTLE chooses the retained family with the largest mean margin, or the reference if none qualifies. At deployment, the score table, selected family and λ remain fixed; Eq. (6) assigns one setting and one generation per prompt without probing alternatives.

Allocator variants. To isolate each component, we start from SETTLE-$\mathrm{G}_{\mathrm{cap}}$ and add one change at a time (Table 2). SETTLE-$\mathrm{G}_{\mathrm{cap}}$ (column $\overline{\mathrm{G}}$) ranks settings by measured calibration accuracy under

a per-prompt cost limit, SETTLE-E_{cap} ($\overline{E}$) evaluates this score with the settling estimate of Eq. (5), SETTLE-E (E) evaluates the per-prompt limit with the average budget of Eq. (6), and SETTLE-H (H) adds cross-validation, which is the reported method.

Training objective. We self-distil Ouro-1.4B to answer within a token budget specified in its prompt. Targets use the longest correct, naturally completed four-loop chain within the budget, when available, followed by its answer. Training updates rank-16 LoRA adapters ($\alpha = 32$) across all 24 layers with frozen base weights. Five controls test these choices: *Uniform sampling* draws budgets and loop counts uniformly; *No budget instruction* removes the instruction; *Feasible budgets only* excludes examples without a correct chain within budget; and *Plain SFT* and *Plain SFT with mixed loops* train on full chains without budget instructions (Appendix F).

5 Experiments

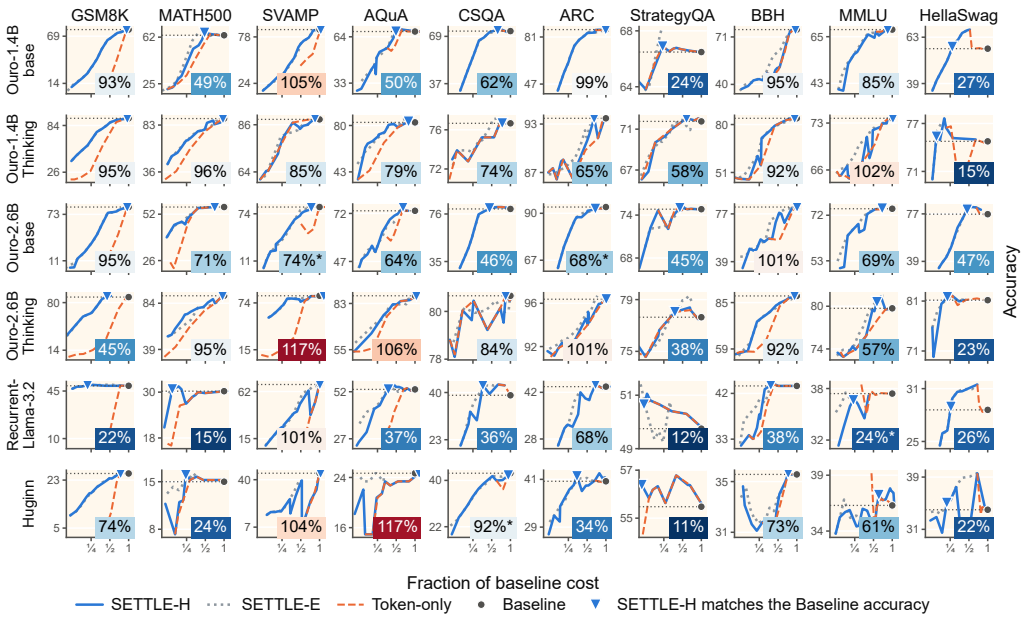

Figure 2: Accuracy against compute for all 6 models $\times$ 10 tasks. Triangles mark where SETTLE-H reaches Baseline accuracy, within one point in starred panels. Boxes report its compute there as a percentage of baseline cost, blue below 100% and red above.

5.1 Setup

Models and tasks. We evaluate six looped models from three families: the base and Thinking variants of Ouro-1.4B and Ouro-2.6B (Zhu et al., 2025), Huginn (Geiping et al., 2025), and Recurrent-Llama-3.2 (McLeish et al., 2025). The evaluation covers ten benchmarks in three groups: math word problems (GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), SVAMP (Patel et al., 2021), and AQuA (Ling et al., 2017)), reasoning (StrategyQA (Geva et al., 2021) and BBH (Suzgun et al., 2022)), and knowledge choice (CommonsenseQA (Talmor et al., 2018), ARC (Clark et al., 2018), MMLU (Hendrycks et al., 2020), and HellaSwag (Zellers et al., 2019)). For each model and task, we generate responses using few-shot prompting and greedy decoding. Allocation policies are evaluated from the stored responses at their selected settings.

Budgets and cost. We evaluate 25%, 50%, 75%, and 100% of the baseline cost, using prefix caching unless stated otherwise (Section 2). Questions with no feasible setting for a policy are scored as incorrect. Reported costs exclude calibration; its cost appears separately in Appendix E.3. For a task with N questions, the calibration set contains $\max(100, \min(300, \lfloor 0.2N \rfloor))$ questions and is disjoint from the evaluation set. Unless stated otherwise, confidence intervals use 400 bootstrap resamples

with the fitted policy fixed. Method comparisons resample the same questions for both methods, whereas averages across model and task combinations resample those combinations (Appendix B.4).

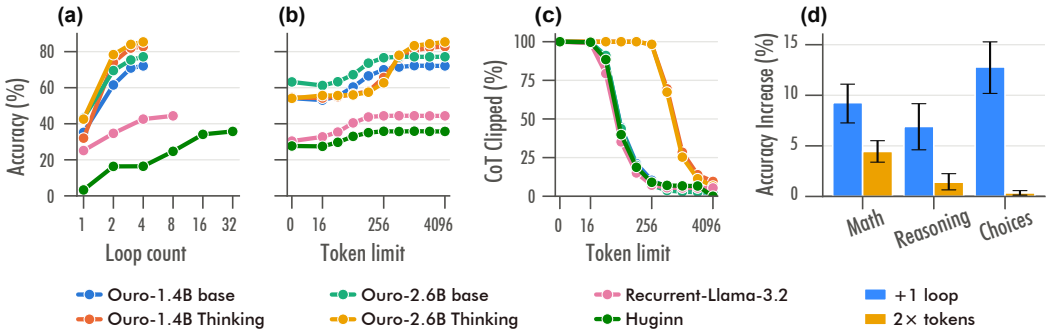

Figure 3: (a) Accuracy against loop count. (b) Accuracy against token limit. (c) CoT clipped share. (d) Points from one more loop and a doubled limit, 49 model-task cases.

Baselines. We compare SETTLE-H with five policies. *Baseline* uses the largest measured loop count and runs to a natural stop unless the per-prompt budget truncates generation (Zhu et al., 2025; Geiping et al., 2025; McLeish et al., 2025); even the full baseline budget truncates chains above the mean baseline cost. *Token-only* keeps this loop count and uses the largest chain-of-thought token limit meeting the average budget, following budget forcing (Muennighoff et al., 2025). *Loop-only* permits the full chain and uses the largest loop count within budget (Geiping et al., 2025). At 25% and 50% of baseline cost, Token-only is feasible for 30 and 44 of 60 model and task combinations, respectively, and Loop-only for 6 and 34. *Answer convergence* stops after a specified number of consecutive extracted answers agree (Liu & Wang, 2025). On the stored probe grid, calibration selects the agreement count and maximum token limit, and all probes are charged. *Best single setting* applies to all task questions the loop count and token limit with the highest calibration accuracy within the average budget.

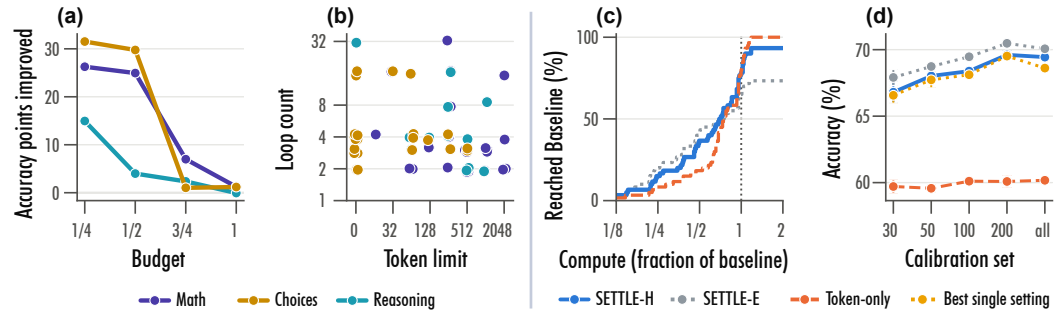

Figure 4: (a) Points of SETTLE-H over Token-only by group. (b) Best single setting at half the cost. (c) Share of cases at Baseline accuracy. (d) Accuracy against calibration size, 49 model-task cases.

5.2 Allocation Under a Budget

Findings. At half the baseline cost, SETTLE-H is 0.8 percentage points $[0.1, 1.5]$ more accurate than the Best single setting; this difference increases to 3.9 points at a quarter of the cost. Both outperform the Baseline, Token-only, and Loop-only at these budgets (Table 2). Part of this difference reflects questions that these policies cannot answer within the budget. However, SETTLE-H remains 9.8 points $[5.5, 15.3]$ more accurate across the 44 model–task cases where Token-only is feasible at half the cost. Across task groups, the improvement over Token-only is smallest on Reasoning. The Best single setting selects a median of 3 loops and a 512-token limit for math word problems, compared with 4 loops and no chain for knowledge-choice tasks (Figure 4(a,b)). SETTLE-H reaches baseline accuracy at a median of 68% of the baseline cost, allowing a one-point tolerance in four cases, compared with 74% for Token-only (Figures 2 and 4(c)).

Table 2: Accuracy (%) across six models and ten tasks at four compute budgets. Costs assume prefix caching and exclude calibration (Appendix E.3); unanswered questions count as incorrect. Baseline is capped per prompt. SETTLE components are added from left to right. Dark and pale blue indicate the best and second-best results; confidence intervals appear in Table 6

		Single axis			Stop rule	Per task	SETTLE			
Model	Budget	Baseline[a]	Token-only[b]	Loop-only[c]	Answer conv.[d]	Best single	$\overline{\text{G}}$	$\overline{\text{E}}$	E	H
Ouro-1.4B base	1/4	13.6	9.9	2.5	9.5	38.5	40.7	41.1	50.8	48.6
	1/2	35.4	31.2	24.0	22.5	66.5	65.7	65.7	68.7	67.7
	3/4	60.9	63.8	52.3	40.7	71.8	71.2	71.2	72.4	72.1
	1	69.2	71.2	71.2	66.7	73.2	73.1	73.1	73.2	73.1
Ouro-1.4B Thinking	1/4	57.8	60.2	0.0	61.7	69.4	69.4	69.5	71.4	71.8
	1/2	73.9	74.2	5.5	73.8	79.0	78.9	78.8	80.1	79.3
	3/4	80.8	80.9	77.9	80.5	82.3	82.1	82.0	82.4	81.2
	1	82.8	82.9	82.9	82.5	82.5	82.6	82.7	82.6	83.2
Ouro-2.6B base	1/4	15.7	11.0	0.0	3.0	47.6	49.2	49.2	55.6	53.4
	1/2	38.3	33.9	39.7	24.2	74.1	71.5	71.3	75.2	75.8
	3/4	62.1	69.2	69.7	44.7	76.0	75.9	75.8	76.9	77.5
	1	74.7	77.1	77.2	71.9	77.4	77.2	77.2	77.3	78.3
Ouro-2.6B Thinking	1/4	54.0	58.2	0.0	60.9	73.4	74.0	74.0	73.1	75.3
	1/2	68.8	68.0	30.2	70.7	81.8	82.0	82.2	83.1	82.6
	3/4	81.7	81.7	82.5	79.6	83.8	83.8	84.3	84.4	84.3
	1	84.9	85.1	84.3	84.2	85.0	84.9	85.5	85.8	85.1
Recurrent-Llama-3.2	1/4	13.6	11.1	4.6	10.5	36.8	35.8	35.9	37.3	36.6
	1/2	24.1	21.4	16.4	19.8	41.8	42.7	43.1	44.0	42.4
	3/4	37.5	41.9	37.8	26.2	44.1	44.0	44.4	44.9	41.9
	1	42.9	42.5	43.6	41.4	44.1	44.5	44.9	45.0	44.5
Huginn	1/4	11.8	8.4	2.9	8.1	26.4	27.8	27.8	30.6	29.7
	1/2	18.8	16.4	20.9	12.7	33.3	33.7	33.7	34.9	33.4
	3/4	29.6	32.7	33.8	20.9	35.4	35.1	34.7	35.4	35.3
	1	34.4	35.1	34.7	32.8	35.5	35.6	35.2	35.4	35.2
All	1/4	27.8	26.5	1.7	25.6	48.7	49.5	49.6	53.1	52.6
	1/2	43.2	40.9	22.8	37.3	62.8	62.4	62.5	64.4	63.5
	3/4	58.7	61.7	59.0	48.8	65.6	65.3	65.4	66.1	65.4
	1	64.8	65.6	65.6	63.3	66.3	66.3	66.4	66.6	66.6

[a] Zhu et al. (2025); Geiping et al. (2025); McLeish et al. (2025) [b] Muennighoff et al. (2025) [c] Geiping et al. (2025) [d] Liu & Wang (2025)

Attribution. The four SETTLE variants in Table 2 introduce settling-based estimation, budget sharing, and cross-validation (Appendix E.1). Replacing per-prompt limits in SETTLE-E_{cap} with an average budget in SETTLE-E provides the largest improvement: 3.5 and 1.9 percentage points at a quarter and half the baseline cost. Adding cross-validation create SETTLE-H, which reduces mean accuracy but limits losses relative to Baseline. At half cost with prefix caching, it is 0.5 points less accurate than SETTLE-E (difference -0.5; 95% CI $[-1.1, 0.1]$) across the 44 cases where Token-only is feasible. Under generation-only costs at the same budget, validation reduces the share of cases more than one point below Baseline from 18% to 2% across the initial 49 cases.

5.3 Loops Against Tokens

Findings. More loops improve average task accuracy in all six models. At the largest loop count, increasing the token limit from 512 to 4096 changes accuracy by less than 0.7 points in four models, though both Thinking variants continue to improve (Figure 3(a,b)). Most chains are truncated at shorter limits; 32 tokens cuts off 79%–100% across models at their largest loop counts (Figure 3(c)). We compare depth and chain length through the accuracy gains from the next measured loop count and a doubled token limit. Across 49 model and task combinations, these gains average 10.2 and 2.2 points, respectively, with depth larger in every task group (Figure 3(d)).

5.4 Training With a Stated Budget

Setup. We train LoRA adapters for Ouro-1.4B at four loops, with the token budget stated in the prompt (Section 4). Table 3 compares the base model with six trained variants at budgets of 32, 512, and 2048 tokens. All variants use the same adapter configuration (Hu et al., 2022) and the model's own correct chains (Zelikman et al., 2022). Plain SFT uses full chains without a budget instruction, whereas Budget-conditioned includes the budget in the prompt, following Aggarwal &

Table 3: Accuracy (%) of Ouro-1.4B at four loops for the base model and six trained variants under 32-, 512-, and 2048-token budgets. Blue and red show gains and losses from the base within each column. Bottom: SETTLE-H for the base and Budget-conditioned, uniform-loops models at 25%, 50%, and 75% of the base model's baseline cost (Appendix F).

	Budget 32 tokens				Budget 512 tokens				Budget 2048 tokens			
Variant	GSM8K	CSQA	AQuA	MATH500	GSM8K	CSQA	AQuA	MATH500	GSM8K	CSQA	AQuA	MATH500
Base model	18.9	72.5	57.1	23.0	76.9	72.3	68.2	56.5	76.9	72.3	67.5	64.0
Budget-conditioned	55.0	74.2	57.1	30.5	78.8	74.9	67.5	55.0	78.9	74.8	66.9	59.8
Uniform sampling	54.4	74.7	60.4	28.0	79.7	74.8	69.5	60.0	80.3	75.2	69.5	68.0
Feasible budgets only	53.3	74.2	61.0	29.5	78.8	74.0	68.2	56.2	78.9	74.1	69.5	62.7
No budget instruction	37.2	75.8	61.7	25.5	73.3	75.6	66.9	45.8	73.3	75.6	67.5	45.2
Plain SFT	15.6	74.1	55.2	22.8	79.0	74.0	67.5	60.8	78.9	74.0	67.5	67.8
Plain SFT, mixed loops	12.4	74.6	55.2	22.5	78.3	75.0	68.8	55.2	78.3	75.0	68.2	66.2
	SETTLE-H, $\frac{1}{4}$ of the base cost				SETTLE-H, $\frac{1}{2}$				SETTLE-H, $\frac{3}{4}$			
Base model	11.6	72.1	57.1	31.8	64.9	73.4	68.8	65.5	74.3	73.4	68.8	64.0
Budget-conditioned, uniform loops	30.7	77.3	56.5	47.0	71.4	77.7	69.5	54.9	78.3	77.6	69.5	57.0

Welleck (2025). The median half-width of the 95% confidence interval is 2 points for GSM8K and CSQA, 4 for MATH500, and 6 for AQuA.

Findings. Table 3 shows three trends. First, budget conditioning helps when the answer fits in a short chain: at 32 tokens, it adds 36 points on GSM8K and 8 on MATH500, while CSQA requires no chain and varies by at most 2.6 points across the three budgets. Second, the budget instruction preserves long-budget accuracy. Variants trained only on budget-fitting chains perform similarly at short budgets, but at 2048 tokens, the no-instruction variant trails the base model by 4 points on GSM8K and 19 on MATH500 because it shortens all chains, not only budgeted ones. Third, the budget-conditioned model's largest loss, 4 points on MATH500 at 2048 tokens, disappears when training budgets are sampled uniformly, pointing to target selection rather than budget-conditioned training. Under SETTLE-H at half the base model's cost, the trained model gains 6.5 points on GSM8K and 4.3 on CSQA but loses 10.6 on MATH500 (Table 3, bottom), consistent with earlier commitment confirmed by the internal-state analysis Appendix G.

5.5 Why It Works

Settling bounds the improvement. The settled share on the calibration questions at the largest loop count, averaged across token limits, predicts the improvement over Token-only. Among the feasible cases in the first 49 model–task cases, its Spearman correlation with the gain is -0.54 at half the baseline cost and -0.69 at a quarter. The correlation remains negative when any single task is excluded (Appendix E.6). Intuitively, when answers already settle early at the largest loop count, per-task allocation has less room to improve accuracy.

Calibration size and cost. Figure 4(d) reports results on the first 49 model–task cases. At half the baseline cost, SETTLE-H reaches 66.8% accuracy with 30 calibration questions, compared with 69.4% using the full calibration set. With 10 labelled questions, the settling estimator is 1.9 points $[0.9, 3.0]$ more accurate than directly looking up the same settings. This advantage falls to 0.7 points with either 20 or 30 questions, and the two methods agree from 50 questions onward (Table 8). The calibration grid costs the equivalent of a median of 839 baseline queries per model–task case, or 734 under the full-prompt cost scheme, and is amortised after a median of 5,565 queries (Table 9).

Inside the model. At a 128-token limit, Ouro-1.4B's state answer matches its full-CoT answer on 88% of GSM8K questions after four loops, versus 55% after one (Figure 5(a)). At the settle point, KL divergence from the full-CoT answer distribution decreases, and full-CoT correctness is predicted better than by Ouro's exit gate on three of four tasks; on MATH500, the AUROCs are 0.73 and 0.50, respectively (Figure 5(b,c)). In the Budget-conditioned, uniform loops model, removing any single adapter module reduces GSM8K accuracy by at most 3.3 points at 32 tokens, while removing the entire adapter from the first six layers reduces it by 10.3 points (Figure 5(d); Appendix G). These analyses use only Ouro-1.4B and 300 questions per task; Appendix E.6 provides cross-model settled-share evidence.

Robustness. Across threshold, split, stopping, horizon, cascade, live-generation, and budget-compliance checks, SETTLE-H's closest alternative is agreement stopping, which is 0.8 points less

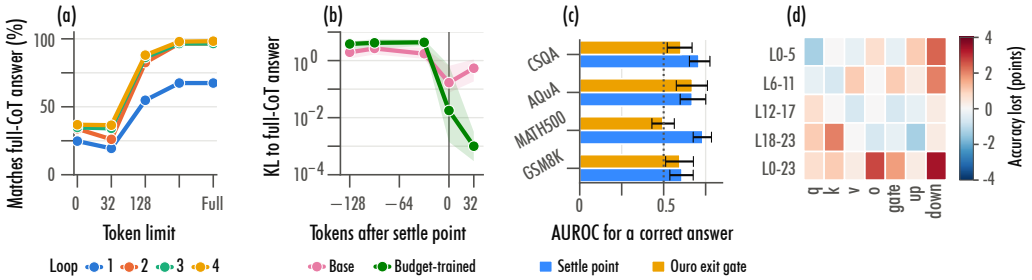

Figure 5: Ouro-1.4B, four loops. (a) State answers equal to the full-CoT answer. (b) KL after the settle point. (c) AUROC for a correct answer. (d) Accuracy lost per removed module.

accurate at half the cost, while live generation completes all 200 runs and matches stored accuracy within 0.1 points on average (Table 10, Appendix E.4).

6 Related Work and Limitations

Prior work on looped models, including Ouro, Huginn, and Recurrent-Llama-3.2 (Zhu et al., 2025; Geiping et al., 2025; McLeish et al., 2025), generally studies loop count and chain length separately. Related studies compare looping with written reasoning (Saunshi et al., 2025; Xu & Sato, 2025; Zhang, 2026; Lu et al., 2025), examine latent computation during pretraining (Huang et al., 2026), or fix the loop count during training (Jeddi et al., 2026; Gao et al., 2026; Schwethelm et al., 2026b). Budgeted-reasoning methods instead control chain length at fixed depth through budget forcing, length-conditioned training, per-question or per-phase budgets, and test-time strategy selection (Muennighoff et al., 2025; Aggarwal & Welleck, 2025; Han et al., 2024; Xu et al., 2025; Snell et al., 2024; Lin et al., 2026). Per-input allocators also count only token cost (Damani et al., 2024; Huang et al., 2025), while stopping rules terminate chains during generation (Yang et al., 2025; Sun et al., 2025; Iwase et al., 2026). Adaptive-depth methods instead vary loops: Ouro learns when to exit, Huginn exits zero-shot (Zhu et al., 2025; Geiping et al., 2025), confidence-based halting can improve these exits (Popescu et al., 2026), and routing methods vary recursion depth or LoRA branches (Bae et al., 2025; Zhang et al., 2026). Serving systems make per-token depth practical (Schwethelm et al., 2026a). Prior work tunes depth or tokens ; SETTLE chooses both under one budget of block passes, without any training.

Limitations. We evaluate only six looped models, four from the Ouro family, with no non-looped or larger models. At full baseline cost, Huginn and Recurrent-Llama-3.2 reach 34.4% and 42.9% accuracy, so Table 2 reports each model separately. The settled-share and robustness analyses cover five models, and the internal-state analysis in Figure 5 covers only Ouro-1.4B. Calibration requires task-specific queries: across 35 model–task cases, the median break-even point ranges from 1,159 queries with 30 calibration questions to 10,152 with the mean calibration size of 277. Settling is observed only at measured token limits within the measured horizon. Training is limited to LoRA adapters for Ouro-1.4B base, and only two variants use a second seed.

7 Conclusion

Loop count and chain length are not interchangeable compute allocations in looped models. Adding a loop improves accuracy more than doubling chain length, though longer chains yield more accuracy per unit of compute until truncation becomes rare. Label-free answer settling enables SETTLE, which selects a loop count and token limit per prompt under an average budget and matches default-setting accuracy at a median of 68% of its cost. Answers stabilise internally at the settle point even when the exit gate misses it. Budget-conditioned training moves settling earlier, helping when short chains suffice but reducing accuracy otherwise. Joint per-question selection remains open, and our results cover only looped models with at most 3.5B parameters.

AI USE STATEMENT

Generative AI was used for assistance with code scripting and drafting initial prose, which was subsequently edited and refined by the authors. Generative AI was not used for data generation, model architecture design, training, or mathematical derivations. The authors independently verified all code, figures, numeric values, and citations, and assume complete responsibility for the final manuscript.

ETHICS STATEMENT

Our study uses only publicly available models and benchmarks and involves no human subjects, personal data, or conflicts of interest. Because the proposed method reduces inference costs without introducing new capabilities, we identify no risks beyond those associated with the underlying models.

REPRODUCIBILITY STATEMENT

All models and benchmarks are public. Appendix A gives the proofs, Appendix B the generation and scoring protocol, Appendix F the training details. The supplementary material contains the code, a sample grid, and a README mapping every exhibit to its script. Full grids and adapters are released after review.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2503.04697.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025.

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. 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. URL https://arxiv.org/abs/2110.14168.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of LM computation. *arXiv preprint arXiv:2410.04707*, 2024.

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.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *arXiv preprint arXiv:2101.02235*, 2021. URL https://arxiv.org/abs/2101.02235.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. URL https://arxiv.org/abs/2009.03300.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jenny Y. Huang, Mehul Damani, Yousef El-Kurdi, Ramon Astudillo, and Wei Sun. Latency and token-aware test-time compute. *arXiv preprint arXiv:2509.09864*, 2025.

Zeyi Huang, Xuehai He, Yong Jae Lee, and Yelong Shen. Trading depth for time in recurrent transformers. *arXiv preprint arXiv:2609.21605*, 2026.

Naoto Iwase, Yuki Ichihara, Mohammad Atif Quamar, and Junpei Komiyama. Reliable chain-of-thought via prefix consistency. *arXiv preprint arXiv:2605.07654*, 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 (ICLR)*, 2026. arXiv:2602.11451.

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. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. Plan and budget: Effective and efficient test-time scaling on reasoning large language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2505.16122.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation : Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017. URL https://arxiv.org/abs/1705.04146.

Xin Liu and Lu Wang. Answer convergence as a signal for early stopping in reasoning. *arXiv preprint arXiv:2506.02536*, 2025.

Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer. *arXiv preprint arXiv:2507.02199*, 2025.

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.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021. URL https://arxiv.org/abs/2103.07191.

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.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2502.17416.

Kristian Schwethelm, Daniel Rueckert, and Georgios Kaissis. Depth-adaptive inference of looped language models via continuous depth batching. *arXiv preprint arXiv:2608.09444*, 2026a.

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

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Renliang Sun, Wei Cheng, Dawei Li, Haifeng Chen, and Wei Wang. Stop when enough: Adaptive early-stopping for chain-of-thought reasoning. *arXiv preprint arXiv:2510.10103*, 2025.

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. *arXiv preprint arXiv:2210.09261*, 2022. URL https://arxiv.org/abs/2210.09261.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018. URL https://arxiv.org/abs/1811.00937.

Kevin Xu and Issei Sato. To cot or to loop? a formal comparison between chain-of-thought and looped transformers. *arXiv preprint arXiv:2505.19245*, 2025. latest version arXiv:2509.25239.

Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*, 2025.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019. URL https://arxiv.org/abs/1905.07830.

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

Xiaojing Zhang, Haifeng Wu, Gang He, Jiyang Shen, Bochen Lyu, and Zhanxing Zhu. MoDr: Mixture-of-depth-recurrent transformers for test-time reasoning. In *International Conference on Learning Representations (ICLR)*, 2026. OpenReview id 9Pba4rcQbE.

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. v1 2025-10-29, v5 2026-07-01.

CONTENTS OF THE APPENDIX

A Proofs **14**
A.1 Proofs of the Main-Text Statements 14
A.2 Selection Under Estimation Error 15
A.3 The Budget Multiplier 16
B Experimental Protocol **16**
B.1 Models and Loop Counts 16
B.2 Tasks, Prompts and Cost measures 17
B.3 Parse Rules 17
B.4 Calibration and Evaluation Splits 17
B.5 Policies and Upper Bounds 17
C Measured Accuracy Grids **18**
C.1 Forced Continuation 18
D Allocation Results **21**
D.1 Per-Task Accuracy at Half the Baseline Cost 21
D.2 Accuracy Differences and Upper Bounds 22
E Ablations and Robustness **22**
E.1 Allocator Components 22
E.2 Label Count 22
E.3 Calibration Cost and Break-Even 24
E.4 Robustness Checks 24
E.5 Loop Count Against Token Limit 24
E.6 Settle Time and Settled Share as Signals 24
F Training Details and Results **25**
F.1 Trained Variants Under a Budget 25
F.2 Controls 25
F.3 Seed Replicate 25
G Probing and Intervening on Internal Representations **26**

A PROOFS

This appendix proves the accuracy decompositions and the change-of-setting identity in Section 3, the selection bound in Section 3.4, and the monotonicity and feasibility claims for the budget multiplier in Section 4.

Throughout, $\mathcal{T} = \{T_0 < \cdots < T_{\max}\}$ is the grid of measured token limits for loop count k. All quantities follow the definitions in Section 3. Every token limit belongs to $\mathcal{T}$, and sums over $j \le T$ run over measured limits. The proofs use the following assumptions and convention:

(i) The grid $\mathcal{T}$ is finite, so $J_{i,k} \in \mathcal{T} \cup \{\infty\}$.

(ii) Conditional means over events of probability zero are assigned the value zero.

(iii) For each question i, correctness depends only on the scored answer: there is a function y_i such that $Y_{i,k}(T) = y_i(r_{i,k}(T))$ for every k and T. Answers that do not parse receive a score of zero.

Lemma 1 and Proposition 1 do not require (iii). Lemma 2 does.

A.1 PROOFS OF THE MAIN-TEXT STATEMENTS

Proof of Lemma 1. The events $\{J_{i,k} \le T\}$ and $\{J_{i,k} > T\}$ partition the questions. Therefore,

$$\begin{aligned} A_k(T) &= \mathbb{E}\big[Y_{i,k}(T)\,\mathbf{1}\{J_{i,k} \le T\}\big] + \mathbb{E}\big[Y_{i,k}(T)\,\mathbf{1}\{J_{i,k} > T\}\big] \\ &= G_k(T)\,\bar{c}_k(T) + [1 - G_k(T)]\,l_k(T). \end{aligned}$$

The second equality follows from the definitions of the conditional means, with convention (ii) covering groups of probability zero.

Proof of Lemma 2. The event $\{J_{i,k} \le T\}$ is the disjoint union of $\{J_{i,k} = j\}$ over measured limits $j \le T$. For a question with $J_{i,k} = j$, Definition 1 gives $r_{i,k}(T') = r_{i,k}(T_{\max})$ at every measured $T' \ge j$. In particular, $r_{i,k}(T) = r_{i,k}(j)$, so assumption (iii) implies $Y_{i,k}(T) = Y_{i,k}(j)$. Hence,

$$\begin{aligned} \mathbb{E}\big[Y_{i,k}(T)\,\mathbf{1}\{J_{i,k} = j\}\big] &= \mathbb{E}\big[Y_{i,k}(j)\,\mathbf{1}\{J_{i,k} = j\}\big] \\ &= p_k(j)\,c_k(j). \end{aligned}$$

Summing over $j \le T$ gives

$$\mathbb{E}\big[Y_{i,k}(T)\,\mathbf{1}\{J_{i,k} \le T\}\big] = \sum_{j \le T} p_k(j)\,c_k(j) = S_k(T).$$

Adding the unsettled contribution from Lemma 1 proves Eq. (2). Questions with $J_{i,k} = \infty$ belong to the unsettled group at every measured limit and require no separate case. The same partition gives $G_k(T) = \sum_{j \le T} p_k(j)$, and comparing the settled term with the proof of Lemma 1 gives $G_k(T)\,\bar{c}_k(T) = S_k(T)$. Thus $\bar{c}_k(T) = S_k(T)/G_k(T)$ whenever $G_k(T) > 0$.

In the experiments, settle times are computed from normalised answers, whereas correctness uses the stored scores of the original answers. Two answers with the same normalised form can therefore receive different scores, violating assumption (iii) when the normalised form is used as $r_{i,k}$. Across the original 49 model-task cases, the resulting discrepancy between the two sides of Eq. (2) is at most 0.20 points.

Proof of Proposition 1. By Lemma 1,

$$A_k(T) = G_k(T)\,\bar{c}_k(T) + U_k(T),$$

and the same identity holds at (k', T'). Subtracting the two identities and adding and subtracting $G_{k'}(T')\,\bar{c}_k(T)$ gives

$$\begin{aligned} A_{k'}(T') - A_k(T) = &\big[G_{k'}(T') - G_k(T)\big]\,\bar{c}_k(T) \\ &+ G_{k'}(T')\,\big[\bar{c}_{k'}(T') - \bar{c}_k(T)\big] \\ &+ U_{k'}(T') - U_k(T). \end{aligned}$$

When $G_{k'}(T') = 0$, the second term is zero under convention (ii), as specified in the proposition.

This proof applies to changes in loop count, token limit, or both. It uses the conditional-mean definition of $\bar{c}_k(T)$ and does not require assumption (iii). Under (iii), Lemma 2 also permits the representation $\bar{c}_k(T) = S_k(T)/G_k(T)$ when $G_k(T) > 0$.

On the evaluation split of the original 49 model-task cases, the two sides of Eq. (3) agree to within 2.8×10^{-14} points across 3272 changes of setting. Each change moves to the next measured loop count at a fixed token limit or to the next measured token limit at a fixed loop count. For this check, accuracy is computed from the right-hand side of Eq. (2) and settled correctness from $S_k(T)/G_k(T)$, using convention (ii) when $G_k(T) = 0$. The check therefore verifies the implementation of the decomposition, independently of the normalisation discrepancy discussed above.

A.2 Selection Under Estimation Error

Fix a budget X and a finite, nonempty set $\mathcal{U}_X$ of feasible settings, with $M = |\mathcal{U}_X|$. For $u = (k, T)$, write $A(u) = A_k(T)$ and let $\widehat{A}(u)$ denote its estimated accuracy. Consider the uniform error bound

$$\max_{u \in \mathcal{U}_X} \left| \widehat{A}(u) - A(u) \right| \leq \varepsilon. \tag{7}$$

Proposition 2 (Selection Under Estimation Error). *Let* $\hat{u} \in \arg\max_{u \in \mathcal{U}_X} \widehat{A}(u)$.

(a) Under Eq. (7),

$$\max_{u \in \mathcal{U}_X} A(u) - A(\hat{u}) \leq 2\varepsilon.$$

Consequently, $A(\hat{u}) \geq A(u_{\mathrm{ref}}) - 2\varepsilon$ *for every reference setting* $u_{\mathrm{ref}} \in \mathcal{U}_X$.

(b) Suppose $n \geq 1$ *labelled calibration questions are drawn independently from the same distribution,* $A(u)$ *is the population accuracy under that distribution, and* $\widehat{A}(u)$ *is the empirical accuracy on those* n *questions. If* $\mathcal{U}_X$ *is fixed independently of the calibration sample, then, for any* $\delta \in (0, 1)$*, Eq. (7) holds with probability at least* $1 - \delta$ *for*

$$\varepsilon = \sqrt{\frac{\ln(2M/\delta)}{2n}}.$$

Proof. (a) For every $u \in \mathcal{U}_X$, optimality of $\hat{u}$ and Eq. (7) give

$$A(\hat{u}) \geq \widehat{A}(\hat{u}) - \varepsilon \geq \widehat{A}(u) - \varepsilon \geq A(u) - 2\varepsilon.$$

Taking the maximum over u proves the first claim. Choosing $u = u_{\mathrm{ref}}$ proves the second.

(b) For each fixed u, the estimate $\widehat{A}(u)$ is the mean of n independent Bernoulli variables with mean $A(u)$. Hoeffding's inequality gives

$$\mathbb{P}\big(|\widehat{A}(u) - A(u)| > \varepsilon\big) \leq 2\exp(-2n\varepsilon^2).$$

A union bound over the M settings yields

$$\mathbb{P}\left(\max_{u \in \mathcal{U}_X} |\widehat{A}(u) - A(u)| > \varepsilon\right) \leq 2M\exp(-2n\varepsilon^2).$$

Substituting the stated value of ε makes the right-hand side equal to δ. Independence between settings is not required, so the estimates may use the same calibration questions.

The factor 2 in part (a) is sharp. For $0 < \varepsilon \leq 1/2$ and $0 \leq a \leq 1 - 2\varepsilon$, consider two settings with

$$A(u_1) = a + 2\varepsilon, \qquad A(u_2) = a, \qquad \widehat{A}(u_1) = \widehat{A}(u_2) = a + \varepsilon.$$

Both estimates satisfy Eq. (7), u_2 is a maximiser, and selecting it incurs a loss of 2ε.

Part (a) applies to any estimator satisfying the uniform error bound, including Eqs. (4) and (5). Part (b) establishes such a bound for the empirical accuracy. It does not establish the same bound for the settling estimators. If the feasible set is selected using the calibration sample, the uniform bound must instead cover a fixed set containing all candidate settings. The proposition concerns selection of a single setting and does not cover the prompt-dependent policy in Eq. (6) or guarantee its cost on evaluation prompts. For $\delta = 0.05$ and $M = 40$, the full grid of an Ouro model, part (b) gives $2\varepsilon = 38.4$ points at $n = 100$ and 22.2 at $n = 300$. These are worst-case bounds, not measured accuracy differences.

A.3 The Budget Multiplier

Let $\mathcal{F}$ be a finite, nonempty family of settings with finite estimated accuracies and costs. For each calibration prompt i and multiplier $\lambda \geq 0$, let $\pi_\lambda(i)$ be any maximiser in Eq. (6). Write $\bar{C}(\lambda)$ for its mean estimated cost over the finite calibration set.

Lemma 3 (Budget multiplier). *(a) If $0 \leq \lambda_1 < \lambda_2$, then*

$$\widehat{C}_i(\pi_{\lambda_2}(i)) \leq \widehat{C}_i(\pi_{\lambda_1}(i))$$

for every prompt i, regardless of how ties are resolved. Consequently, $\bar{C}(\lambda)$ is nonincreasing.

(b) For all sufficiently large λ, every selected setting has minimum estimated cost for its prompt. Hence, some multiplier satisfies the budget if and only if

$$\frac{1}{n_{\mathrm{cal}}} \sum_{i=1}^{n_{\mathrm{cal}}} \min_{u \in \mathcal{F}} \widehat{C}_i(u) \leq X,$$

where n_{cal} is the number of calibration prompts.

Proof. (a) Fix a prompt i and write $u_1 = \pi_{\lambda_1}(i)$ and $u_2 = \pi_{\lambda_2}(i)$. Optimality at the respective multipliers gives

$$\widehat{A}_{\mathrm{res}}(u_1) - \lambda_1 \widehat{C}_i(u_1) \geq \widehat{A}_{\mathrm{res}}(u_2) - \lambda_1 \widehat{C}_i(u_2),$$
$$\widehat{A}_{\mathrm{res}}(u_2) - \lambda_2 \widehat{C}_i(u_2) \geq \widehat{A}_{\mathrm{res}}(u_1) - \lambda_2 \widehat{C}_i(u_1).$$

Adding these inequalities yields

$$(\lambda_2 - \lambda_1)\big[\widehat{C}_i(u_1) - \widehat{C}_i(u_2)\big] \geq 0.$$

Since $\lambda_2 > \lambda_1$, the selected cost cannot increase. Averaging over prompts proves the claim for $\bar{C}$.

(b) Let

$$\Delta = \max_{u \in \mathcal{F}} \widehat{A}_{\mathrm{res}}(u) - \min_{u \in \mathcal{F}} \widehat{A}_{\mathrm{res}}(u).$$

Fix a prompt i and let u_0 be a minimum-cost setting. If all settings have the same cost, every maximiser already has minimum cost. Otherwise, finiteness of $\mathcal{F}$ ensures that

$$\gamma_i = \min_{\substack{u \in \mathcal{F} \\ \widehat{C}_i(u) > \widehat{C}_i(u_0)}} \big[\widehat{C}_i(u) - \widehat{C}_i(u_0)\big] > 0.$$

For any setting u with higher cost and any $\lambda > \Delta/\gamma_i$,

$$\big[\widehat{A}_{\mathrm{res}}(u_0) - \lambda \widehat{C}_i(u_0)\big] - \big[\widehat{A}_{\mathrm{res}}(u) - \lambda \widehat{C}_i(u)\big] \geq -\Delta + \lambda \gamma_i > 0.$$

Thus no higher-cost setting can maximise the objective. Since the calibration set is finite, a common finite multiplier ensures minimum-cost selections for every prompt. Every policy incurs at least this mean minimum cost, proving both directions of the feasibility claim.

The mean cost $\bar{C}(\lambda)$ is a nonincreasing step function. If ties in Eq. (6) are resolved in favour of a lower-cost setting, it is also right-continuous. For a feasible budget, the multipliers satisfying $\bar{C}(\lambda) \leq X$ therefore form an interval $[\lambda^*, \infty)$. Once a feasible upper endpoint is found, bisection approximates the smallest feasible multiplier λ^*.

B Experimental Protocol

B.1 Models and Loop Counts

Loop counts. We evaluate Ouro at $k \in \{1, 2, 3, 4\}$, Recurrent-Llama-3.2 at $\{1, 2, 4, 8\}$, and Huginn at $\{1, 2, 4, 8, 16, 32\}$ on all ten tasks. Recurrent-Llama-3.2 is additionally evaluated at 16 and 32

loops on GSM8K. Its baseline therefore uses 32 loops on GSM8K and 8 elsewhere. Each chain is generated until the model stops.

Output extraction. Ouro-2.6B's released exit threshold reads about 1% of tokens after the first loop. To ensure that each setting uses the specified depth, all Ouro outputs are read after the final loop.

Evaluation coverage. The allocation tables include all 60 model-task cases. Analyses restricted to the initial 49 cases exclude Huginn and Ouro-2.6B base on SVAMP and are identified explicitly. At half the baseline cost, cross-validation retains no candidate family on seven of Huginn's ten tasks, so SETTLE-H uses the reference policy. The SVAMP case was initially excluded because the 8-token answer reserve could truncate numeric answers. Increasing the reserve to 24 tokens changes baseline accuracy by 1.5 points.

B.2 Tasks, Prompts and Cost measures

Prompts. We use the evaluation code's few-shot prompts, with a shared block of examples for each task and each BBH or MMLU subtask. Prompts are tokenised separately for each model, and these token counts determine the shared-prefix lengths used in prefix-cached costs.

Cost measures. The three measures differ only in the prompt charge Q_i in Eq. (1): generation-only uses $Q_i = 0$, full-prompt uses $Q_i = P_i$, and prefix-cached uses $Q_i = P_i - S_i$. Here, P_i is the prompt length and S_i is the shared few-shot prefix, computed once per task and loop count. Prefix-cached costs are used unless stated otherwise. The shared prefix accounts for about half of the full-prompt baseline cost. All measures include the truncated chain and R_i, the answer-request suffix length plus a task-specific generation reserve. Expected chain lengths are estimated from calibration questions. Costs are measured in block passes, rather than FLOPs or elapsed time. At the full budget, initialising SETTLE-H's cache costs the equivalent of 1.5 baseline queries at the median.

B.3 Parse Rules

Scoring. When a chain is truncated, we append an answer request and allow up to $R_{\mathrm{task}(i)}$ tokens for the response. An incomplete answer is treated as a parse failure. If the answer at the largest token limit does not parse, its settle time is infinite (Definition 1). **Forced continuation.** As a control, we force chains to continue to the generation horizon and extract answers at the same token limits. This protocol is evaluated on four tasks for each Ouro model (Appendix C.1). All allocation results use chains that stop naturally.

B.4 Calibration and Evaluation Splits

Splits. Calibration and evaluation questions are disjoint. For a task with N questions, the calibration set contains $\max(100, \min(300, \lfloor 0.2N \rfloor))$ questions.

Confidence intervals. Unless stated otherwise, we report 95% bootstrap confidence intervals from 400 resamples, holding the fitted policies fixed. Comparisons restricted to questions answered by both policies use the same resampled questions for both. Averages across model-task cases weight cases equally, with intervals obtained by resampling cases. The cross-validation margin test uses 2000 resamples of the held-out questions in each direction of the split.

Estimator bins. The resolved estimator assigns a separate correctness rate to a group only when it contains at least eight labelled questions. Otherwise, it pools token limits into $\{0\}$, $\{16, 32\}$, $\{64, 128\}$, $\{256, 512\}$, and $\{1024, 2048, 4096\}$. Bins with fewer than eight labels use the pooled correctness estimate for their loop count. These pooling rules can make the estimate differ from direct calibration accuracy. The label-count study in Appendix E.2 subsamples only the labels used to estimate correctness. Settling proportions and the margin test retain the full calibration set.

B.5 Policies and Upper Bounds

Budget constraints. The multiplier enforces the average budget on calibration prompts. Evaluation cost may exceed this budget when prompt lengths differ, and selected settings are not adjusted after-

wards. Baseline instead applies a cost limit to each prompt, while Token-only selects a token limit using mean calibration cost. SETTLE-E and SETTLE-H use the multiplier in Eq. (6). SETTLE-H's reference policy is specified in Appendix E.4.

Comparison policies. We evaluate 20 budget fractions up to twice the baseline cost. Baseline uses the largest measured loop count and the largest token limit affordable for each prompt. Even at the full baseline budget, it truncates chains whose cost exceeds the mean baseline cost. Questions with no affordable setting count as incorrect. Token-only uses the largest loop count and the largest token limit satisfying the average budget. Loop-only retains the full chain and selects the largest loop count satisfying that budget. Best single setting selects the setting with the highest calibration accuracy among those satisfying the average budget.

Upper bounds. The per-question upper bound uses evaluation labels to select the cheapest correct setting for each question. Questions answered incorrectly at every setting remain incorrect and are charged at their cheapest setting. This provides an upper bound on accuracy over the measured settings but requires labels at inference. The per-task upper bound selects, for each model-task case and budget, the setting with the highest evaluation accuracy among those whose mean calibration cost satisfies the budget. It is therefore at least as accurate as Best single setting.

C Measured Accuracy Grids

Grids. Figures 6 to 8 give the accuracy at every measured setting under the model's own stop. They are the data for Figure 1 and Section 5.3. Columns run to 8192 where measured, but the allocation results use the 4096-token column for the baseline setting. The Ouro-2.6B base SVAMP panel is the original grid (Appendix B.1).

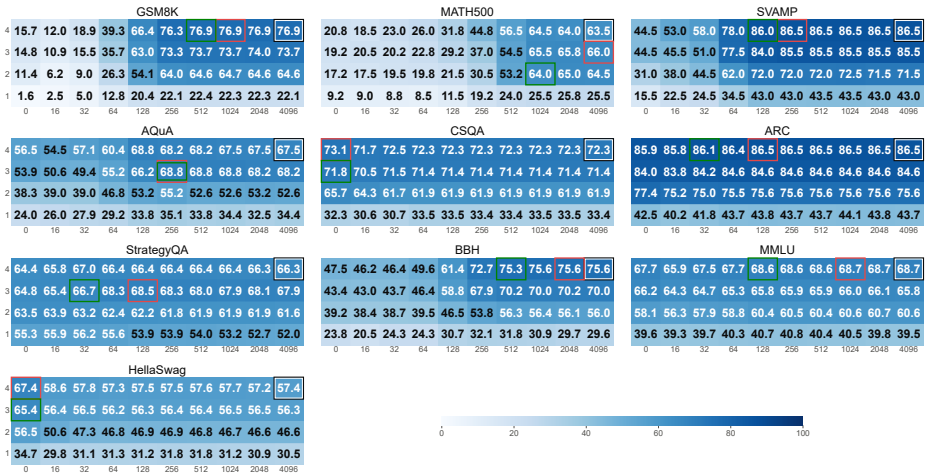

Figure 6: Accuracy (%) by loop count (rows) and token limit (columns) on the ten tasks for Ouro-1.4B base. The ink box marks the baseline setting, the red box the most accurate setting, and the green box the cheapest setting within 0.5 points of the baseline accuracy. Grey entries were not run.

C.1 Forced Continuation

Over 192 comparisons with the model's own stop, forced continuation (Appendix B.3) changes accuracy by less than 0.1 points at a token limit of 128. It lowers accuracy by 1.7 points at 4096, most on Ouro-2.6B Thinking. Figures 9 and 10 give the grids on the four tasks run per model.

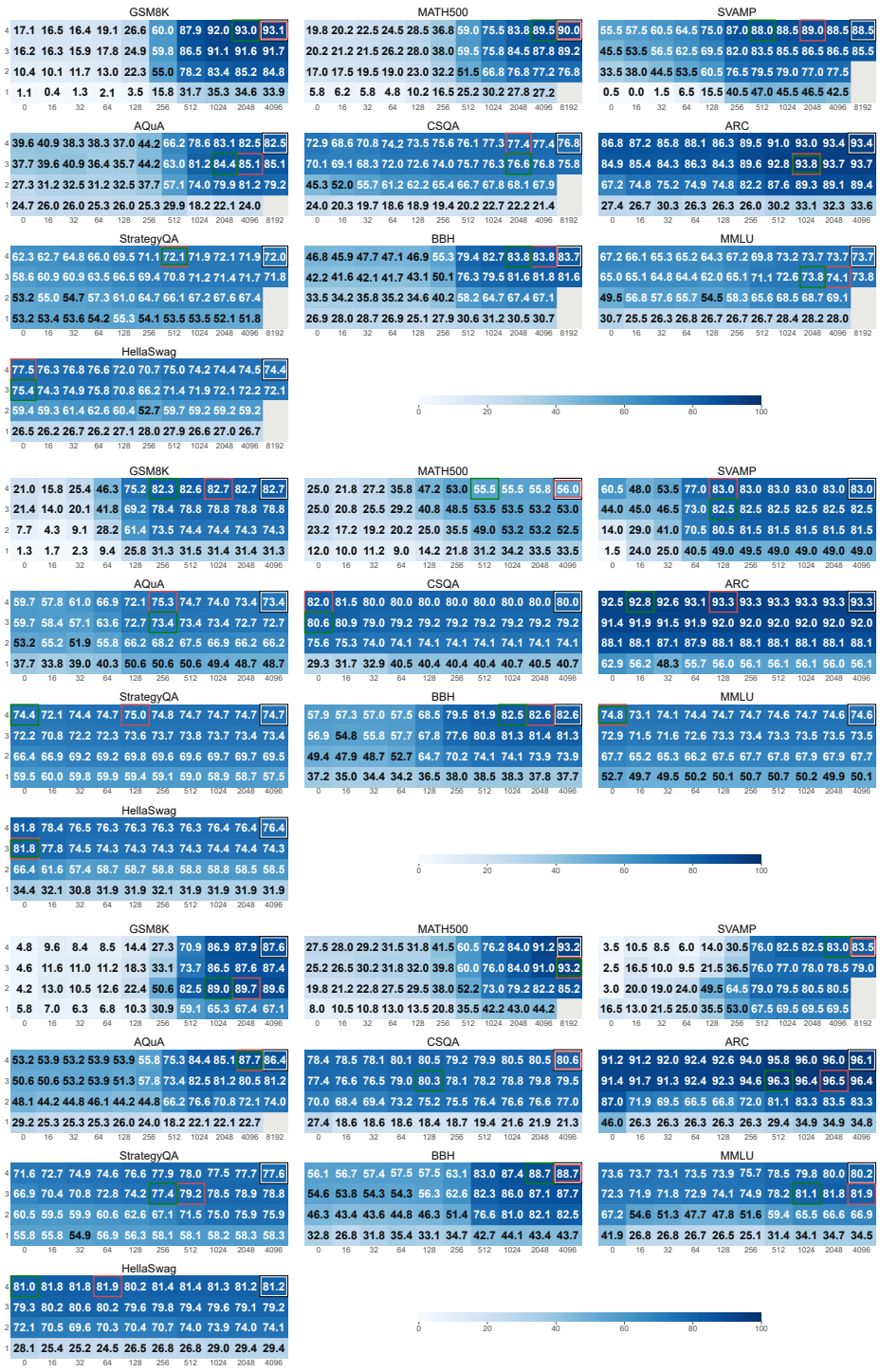

Figure 7: Accuracy (%) by loop count (rows) and token limit (columns) on the ten tasks for Ouro-1.4B Thinking (top), Ouro-2.6B base (middle) and Ouro-2.6B Thinking (bottom). Boxes and grey entries as in Figure 6.

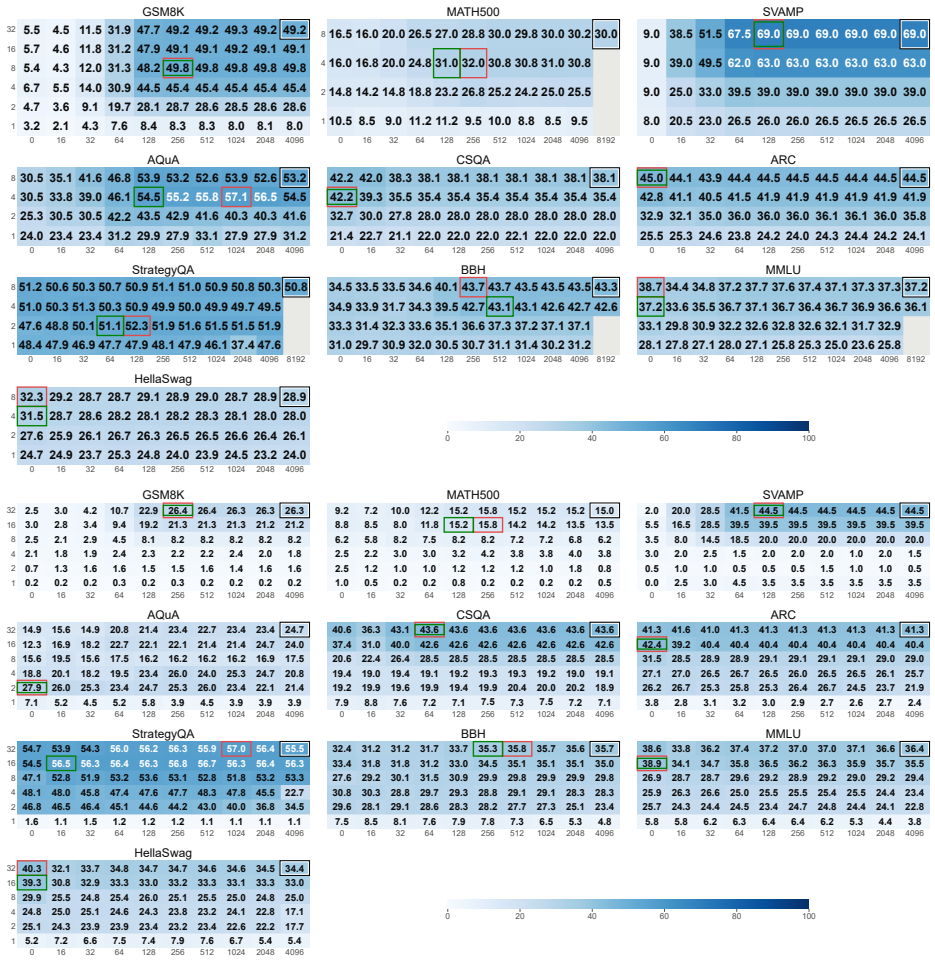

Figure 8: Accuracy (%) by loop count (rows) and token limit (columns) on the ten tasks for Recurrent-Llama-3.2 (top) and Huginn (bottom). Boxes and grey entries as in Figure 6.



Figure 9: Accuracy (%) by loop count (rows) and token limit (columns) under forced continuation on four tasks for Ouro-1.4B base (top two rows) and Ouro-1.4B Thinking (bottom two rows). Boxes and grey entries as in Figure 6.

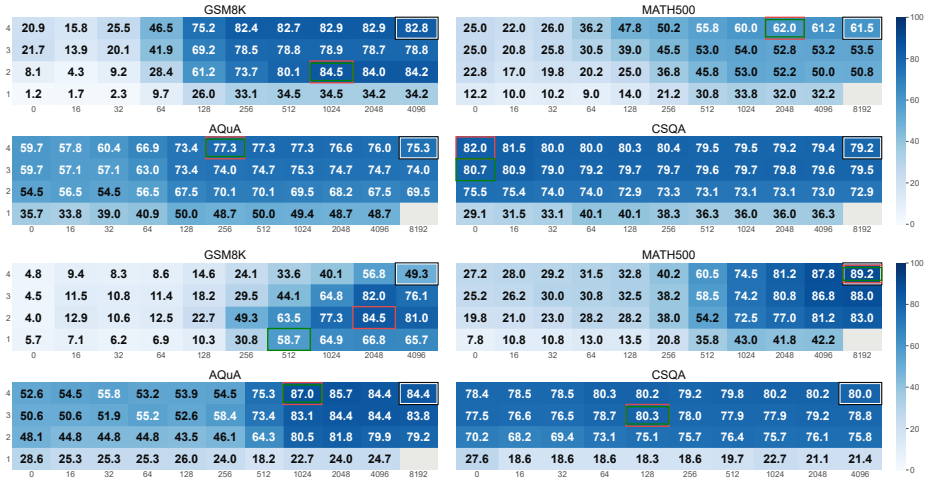

Figure 10: Accuracy (%) by loop count (rows) and token limit (columns) under forced continuation on four tasks for Ouro-2.6B base (top two rows) and Ouro-2.6B Thinking (bottom two rows). Boxes and grey entries as in Figure 6.

D Allocation Results

D.1 Per-Task Accuracy at Half the Baseline Cost

Table 4: Accuracy (%) of Token-only, the Baseline and SETTLE-H on each of the 6 models × 10 tasks at half the baseline cost, prefix-cached, and of Ouro's exit gate on four tasks of Ouro-1.4B base, generation-only. A dash marks a policy with no setting within the budget, and Mean is over the tasks where the row has a value.

Policy	GSM8K	MATH500	SVAMP	AQuA	CSQA	ARC	SQA	BBH	MMLU	Hella.	Mean
Ouro-1.4B base											
Token-only	18.7	56.5	–	57.1	–	–	66.7	45.8	–	66.9	52.0
Baseline	18.8	55.5	32.5	54.5	0.3	15.9	66.6	41.9	46.1	21.6	35.4
SETTLE-H	66.2	65.5	70.0	67.5	72.1	83.7	66.7	50.8	67.4	66.9	67.7
Exit gate	20.3	53.3	–	71.4	74.0	–	–	–	–	–	–
Ouro-1.4B Thinking											
Token-only	59.8	75.5	87.0	66.2	75.2	89.8	71.0	79.3	67.1	70.6	74.2
Baseline	59.8	74.0	87.0	66.2	75.2	88.8	71.0	77.9	67.8	71.1	73.9
SETTLE-H	87.4	78.5	84.0	81.2	75.2	90.9	71.0	79.3	67.8	77.6	79.3
Ouro-2.6B base											
Token-only	16.6	55.5	–	61.0	–	–	74.7	56.6	74.5	–	56.5
Baseline	22.5	54.3	39.0	59.7	0.4	17.6	74.8	40.2	51.2	22.8	38.3
SETTLE-H	75.6	55.5	75.0	71.4	81.0	90.6	74.7	77.9	74.5	82.1	75.8
Ouro-2.6B Thinking											
Token-only	27.6	76.2	30.5	75.3	79.2	92.6	78.0	63.5	75.8	81.2	68.0
Baseline	27.3	73.8	30.5	75.3	79.2	93.1	78.0	74.4	75.2	80.9	68.8
SETTLE-H	89.0	81.8	75.5	82.5	79.2	95.2	78.0	84.0	79.3	81.2	82.6
Recurrent-Llama-3.2											
Token-only	11.4	29.8	–	41.6	–	–	50.4	43.7	37.6	–	35.7
Baseline	14.6	30.2	3.5	47.4	0.1	8.1	50.4	43.1	33.1	10.9	24.1
SETTLE-H	48.9	29.8	49.5	54.5	39.5	40.2	50.4	43.8	36.7	30.7	42.4
Huginn											
Token-only	4.0	15.2	–	22.7	–	–	56.8	31.6	34.1	–	27.4
Baseline	9.8	15.0	1.0	22.7	1.0	7.5	56.8	30.2	29.3	14.5	18.8
SETTLE-H	23.7	15.2	36.0	22.7	38.9	39.4	56.8	32.7	36.2	32.5	33.4

Table 5: Accuracy (%) by model of the Baseline run to its own stop and of the per-question upper bound at the baseline cost, prefix-cached, mean over each model's 10 tasks. Reach is the median compute at which SETTLE-H reaches the Baseline accuracy in % of the baseline cost, with the number of tasks reached, and a task not reached within twice the baseline cost ranks above 200%.

	Baseline,	Per-question	Reach	
Model	own stop	bound	median %	tasks
Ouro-1.4B base	72.1	92.8	74	10
Ouro-1.4B Thinking	82.8	96.4	82	10
Ouro-2.6B base	77.7	92.8	70	8
Ouro-2.6B Thinking	85.4	96.7	88	10
Recurrent-Llama-3.2	44.3	81.5	37	9
Huginn	35.4	77.3	67	9
6 models $\times$ 10 tasks	66.3	89.6	70	56

SETTLE-H is feasible on all 60 model-task cases. Token-only is infeasible on 16 (Table 4).

D.2 Accuracy Differences and Upper Bounds

Compute to reach. Figure 2 reports the cost at the first budget where SETTLE-H reaches baseline accuracy. Four cases do not reach it within twice the baseline cost. Using their first crossing within one point gives the reported median of 68%. All cases reach this tolerance.

Baseline comparisons. Across all 60 cases, SETTLE-H exceeds the capped Baseline by 20.3 points $[14.6, 26.2]$ at half the baseline cost and the uncapped Baseline by 0.3 points $[-0.1, 0.8]$ at full cost (Table 5).

Upper bounds. At half the baseline cost, the per-question bound reaches its unbudgeted accuracy on all 60 cases. On the 44 cases where Token-only is feasible, SETTLE-H closes 28.8% $[16.7, 39.3]$ of the gap from Token-only to this bound and 94.1% $[85.1, 100.0]$ to the per-task bound (Appendix B.5). Intervals use 2000 resamples of cases. By varying settings across prompts, SETTLE-H can exceed the per-task bound, as it does at a quarter of baseline cost.

E Ablations and Robustness

E.1 Allocator Components

Sharing the budget across prompts provides the largest improvement: 3.2 points at a quarter of baseline cost over the initial 49 cases. At half cost, SETTLE-E falls more than one point below the better of Token-only and Baseline on 5 of the 44 cases where Token-only is feasible.

E.2 Label Count

Estimator comparison. The estimator of Eq. (4) uses two correctness rates per loop count. With ten labels, it falls 4.49 points below the per-task bound, compared with 7.20 for direct lookup. At 100 labels, all three estimators perform similarly. With more labels, its approximation bias reduces accuracy. A control on 20 cases at full baseline cost, using per-prompt limits without validation, gives the same ordering at ten labels and comparable lookup and resolved estimates at 100. These comparisons measure label requirements, excluding the compute needed to extract answers.

Settling versus lookup. Settling-based selection improves accuracy with 10 to 30 labels and performs comparably to direct lookup from 50 labels onward (Table 8).

Calibration size. On the 35 cases supporting every calibration size, SETTLE-H exceeds Token-only by 10.6 points with 50 labels and 11.3 with the full set, at half baseline cost under prefix-cached costing. Comparisons average the 25 to 26 cases where Token-only is feasible. With 30 labels, SETTLE-H underperforms SETTLE-E and reverts to the reference on 79% of cases, compared with 57% using the full set.

Table 6: SETTLE-H minus each policy in points at four budgets, prefix-cached, over the 6 models $\times$ 10 tasks (top) and the 49 cases measured first (bottom, with each policy's accuracy). Differences are over the cases where the comparator fits the budget, on the questions both policies answer. Parentheses give the number of such cases and the cases whose 95% interval lies above or below zero. Table 2 instead counts an unfit case as 0.

Policy	0.25×	0.5×	0.75×	1.0×
All 6 models × 10 tasks				
SETTLE-H minus each policy in points (cases; cases above / below zero beyond their interval)				
Token-only	+10.1 (30; 12/0)	+9.8 (44; 21/0)	+3.7 (60; 16/0)	+1.0 (60; 8/1)
Loop-only	+16.1 (6; 5/0)	+21.4 (34; 27/0)	+3.7 (57; 24/3)	+0.9 (60; 8/1)
Best single setting	+3.9 (60; 22/4)	+0.8 (60; 14/4)	-0.2 (60; 5/7)	+0.3 (60; 5/3)
SETTLE-G_{cap}	+2.7 (60; 24/4)	+1.1 (60; 19/4)	+0.1 (60; 13/5)	+0.3 (60; 6/4)
SETTLE-E_{cap}	+2.6 (60; 23/5)	+1.1 (60; 19/3)	-0.0 (60; 12/5)	+0.2 (60; 5/3)
SETTLE-E	-0.6 (60; 5/9)	-0.8 (60; 2/7)	-0.7 (60; 2/7)	+0.0 (60; 3/3)
Baseline coverage, and SETTLE-H minus the Baseline on the questions the Baseline answers (cases)				
Cases defined	45	60	60	60
Cases fully covered ($\geq$ 99.9%)	13	24	29	40
All defined cases	+10.8 (45)	+9.4 (60)	+3.3 (60)	+1.1 (60)
Fully covered cases	+14.0 (13)	+7.8 (24)	+1.1 (29)	+1.6 (40)
The 49 model-task cases measured first, accuracy (%) over the cases where the policy is defined (cases)				
Token-only	55.7 (27)	60.2 (38)	67.9 (49)	71.6 (49)
Loop-only	35.6 (2)	46.3 (25)	67.8 (46)	71.6 (49)
Best single setting	53.7 (49)	68.6 (49)	71.6 (49)	72.2 (49)
Baseline	57.2 (39)	59.9 (49)	68.4 (49)	71.6 (49)
SETTLE-G_{cap}	54.7 (49)	68.2 (49)	71.3 (49)	72.2 (49)
SETTLE-E_{cap}	54.8 (49)	68.2 (49)	71.4 (49)	72.5 (49)
SETTLE-E	58.0 (49)	70.1 (49)	72.0 (49)	72.6 (49)
SETTLE-H	57.8 (49)	69.4 (49)	71.2 (49)	72.7 (49)
SETTLE-H minus each policy in points (cases; cases above / below zero beyond their interval)				
Token-only	+11.1 (27; 11/0)	+10.8 (38; 19/0)	+3.3 (49; 13/0)	+1.0 (49; 6/1)
Loop-only	+3.4 (2; 1/0)	+25.6 (25; 20/0)	+4.3 (46; 22/3)	+1.0 (49; 8/1)
Best single setting	+4.1 (49; 17/3)	+0.8 (49; 12/3)	-0.4 (49; 3/7)	+0.4 (49; 4/2)
SETTLE-G_{cap}	+3.3 (49; 20/3)	+1.2 (49; 17/3)	-0.1 (49; 10/5)	+0.4 (49; 5/3)
SETTLE-E_{cap}	+3.1 (49; 19/4)	+1.2 (49; 17/2)	-0.3 (49; 9/5)	+0.2 (49; 4/2)
SETTLE-E	-0.2 (49; 5/8)	-0.6 (49; 2/5)	-0.8 (49; 1/7)	+0.1 (49; 1/2)
Baseline coverage, and SETTLE-H minus the Baseline on the questions the Baseline answers (cases)				
Cases defined	39	49	49	49
Cases fully covered ($\geq$ 99.9%)	12	22	26	35
All defined cases	+11.9 (39)	+9.8 (49)	+3.0 (49)	+1.1 (49)
Fully covered cases	+15.2 (12)	+8.5 (22)	+1.2 (26)	+1.5 (35)

Table 7: Cases more than one point below Baseline / cases where Baseline is feasible, among the initial 49 model-task cases.

Cost scheme	Variant	0.25×	0.5×	0.75×	1.0×
Full-prompt	SETTLE-G_{cap}	3/6	2/24	4/39	8/49
Full-prompt	SETTLE-E_{cap}	3/6	2/24	2/39	5/49
Full-prompt	SETTLE-E	2/6	1/24	3/39	4/49
Full-prompt	SETTLE-H	3/6	2/24	3/39	1/49
Generation-only	SETTLE-G_{cap}	2/43	5/49	8/49	7/49
Generation-only	SETTLE-E_{cap}	2/43	5/49	7/49	4/49
Generation-only	SETTLE-E	8/43	9/49	7/49	6/49
Generation-only	SETTLE-H	0/43	1/49	1/49	0/49

Table 8: Settling estimate minus direct lookup in points at half baseline cost, prefix-cached, with 95% intervals over cases. Lead counts require a difference above one point. Subsampled results average three draws. 200 labels covers 42 cases, all other rows 60.

Calibration questions	Settling minus lookup	Settling leads	Lookup leads
10	$+1.86$ $[+0.88, +2.99]$	29	7
20	$+0.68$ $[+0.12, +1.25]$	21	8
30	$+0.65$ $[+0.21, +1.11]$	17	8
50	$+0.15$ $[-0.14, +0.48]$	5	4
100	$+0.19$ $[-0.01, +0.41]$	4	1
200	$+0.02$ $[-0.05, +0.09]$	0	0
All	$+0.03$ $[-0.12, +0.19]$	1	1

E.3 Calibration Cost and Break-Even

Prefix-cached calibration costs a median of 839 baseline queries. Savings at the full budget recover this cost after a median of 5,565 queries among the 48 of 49 cases that break even (Table 9).

Table 9: Median calibration cost and break-even workload, in baseline queries. Top: initial 49 cases. Bottom: 35 calibration-size cases at half baseline cost, prefix-cached. Parentheses show the spread of the cost or counts of cases that do not break even.

		Break-even from savings at	
	Calibration cost	Equal accuracy	Full budget
Prefix-cached	839		5,565 (1)
Generation-only	931 (549 to 1,052)	1,638 (13)	4,606
Full-prompt	734 (302 to 864)	2,435 (15)	9,080
30 calibration questions	99	1,159	
All (277 on average)	897	10,152	

E.4 Robustness Checks

Reference and families. Cross-validation uses the most accurate calibration policy among the feasible baseline setting, the largest loop count with an average-budget token limit, and the largest loop count with per-prompt limits. The four candidate families are: the largest loop count with $T = 0$, that loop count with any token limit, the next smaller loop count with any token limit, and all settings.

Table 10: Robustness checks on the initial 49 cases, using generation-only costs unless stated otherwise. Accuracy differences are in points.

Check	Result
Margin threshold	At half cost, thresholds of 0 and 0.5 bootstrap standard deviations both exceed Baseline by 6.3 points, but leave 10 and 1 cases more than one point below it, respectively.
Redrawn splits	Across five splits, SETTLE-H exceeds Baseline by 4.6 to 6.2 points at half cost, versus 6.3 for the original split. It remains more accurate at every tested budget and split.
Stopping on agreement	Stopping when the last two or three extracted answers agree is 0.8 points below SETTLE-H at half cost, using 76% of the budget.
8192-token horizon	Across 15 cases, baseline accuracy is 70.97% at 8192 tokens versus 70.93% at 4096. SETTLE-H reaches 70.98% using 73% of the longer-horizon baseline cost.
Loop-count cascade	Increasing loop counts until answers agree falls 4.86 to 17.55 points below SETTLE-H and 1.9 to 7.8 below Token-only across budgets, under prefix-cached and generation-only costs.
Live generation	All 200 runs completed across 50 cases from five models at half and full cost. Accuracy differs from stored results by +0.07 points overall and −2.6 to +2.6 per run.
Budget compliance	No case exceeds its generation-only budget. Under full-prompt costs, cases exceeding the budget by more than 1% decline from 11 to 2 of 49 as the budget increases. The largest mean excess is 8.9%.

E.5 Loop Count Against Token Limit

Three-term split. On the initial 49 cases, one more loop adds $[8.8, 11.8]$ points and a doubled token limit $[1.5, 2.9]$ (Figure 3(d)). The loop acts through settled correctness, a term of 8.5 points that is positive on all 49 cases. The token limit acts through settling speed, and its settled-correctness term is negative on all 49. **Shares per loop count.** A larger loop count raises accuracy and the settled share at every token limit (Figure 11).

E.6 Settle Time and Settled Share as Signals

Settle time. Correctness differs by settle time, which is why Lemma 2 gives each settle time its own rate. At the largest loop count, earlier settlers are more often correct on all 49 cases, with an AUROC of 0.685 $[0.667, 0.702]$. **Settled share as a predictor.** The settled share at the largest loop count, averaged over token limits and computed without labels, correlates with the SETTLE-H minus Token-only difference: Spearman $[-0.74, -0.30]$ at half the baseline cost on 38 cases and $[-0.93, -0.32]$ at a quarter on 27. The sign holds with any one task removed. Neither it nor the cost of a chain-free answer predicts the smaller difference over Best single setting ($|\rho| \leq 0.18$).

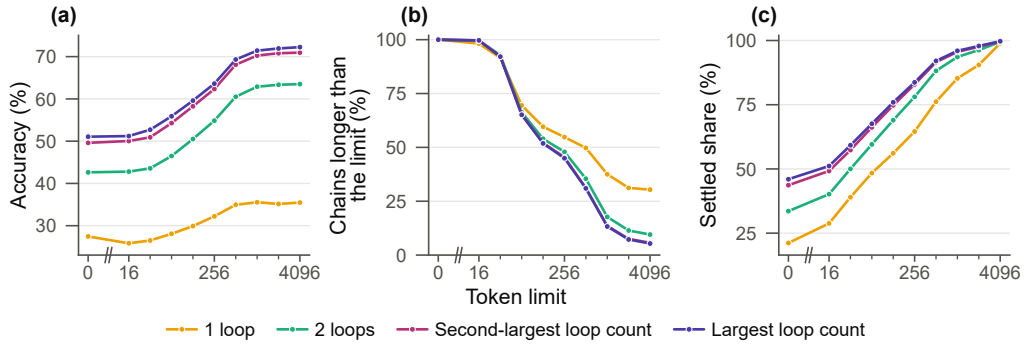

Figure 11: (a) Accuracy, (b) CoT clipped share and (c) settled share against the token limit at four loop counts, mean over the 49 model-task cases measured first. The loop counts are 1, 2 and the two largest of each model.

F Training Details and Results

Procedure. Training examples contain a question, a budget instruction before the answer prefix, and a target selected for that budget. Few-shot examples remain unchanged; removing the instruction recovers the base-model prompt. Each sequence uses one loop count and a length of 1024, 1536, 2048, 2560, or 3072 tokens, with 32 sequences per length per optimiser step. Runs target approximately 1.2M supervised tokens: Budget-conditioned training completed 925 steps and Uniform sampling 669. Each variant took approximately nine hours on one NVIDIA GB10.

Variants. Uniform sampling draws budgets and loop counts uniformly instead of using the base model's grid. No budget instruction removes the instruction during training and evaluation; Feasible budgets only excludes examples without a correct chain within budget. Plain SFT uses full chains at the largest loop count, while Plain SFT with mixed loops uses the Budget-conditioned loop distribution. Two additional controls make only budgets or only loop counts uniform; the latter is the Budget-conditioned, uniform loops variant in Section 5.4.

F.1 Trained Variants Under a Budget

Cost matching. The bottom block of Table 3 uses generation-only costs at fractions f of the base model's baseline cost. For a trained model, we evaluate SETTLE-H at the corresponding fraction $fC_{\text{base}}/C_{\text{trained}}$, where C denotes baseline cost, interpolating linearly between its measured fractions $\{0.25, 0.5, 0.75, 1\}$. These costs differ from the prefix-cached results in Table 4.

Capped Baseline. At half the base model's baseline cost, Budget-conditioned, uniform loops achieves 38.7%, 54.9%, 69.5%, and 77.7% on GSM8K, MATH500, AQuA, and CommonsenseQA using its own capped Baseline.

Budget compliance. On GSM8K at four loops, all trained variants, including Plain SFT, finish within 512 tokens on 99–100% of questions and within 128 tokens on 60–77%.

Long-budget accuracy. On MATH500 at two loops and 2048 tokens, Budget-conditioned training reduces accuracy by 17.0 points relative to the base model; Plain SFT shows no corresponding loss.

F.2 Controls

At 32 tokens, Budget-conditioned training exceeds No budget instruction by 6.7 points on average across six grids (Table 12).

F.3 Seed Replicate

We repeat Budget-conditioned training and Uniform sampling with a second seed to assess variation across runs. Budget-conditioned seeds differ by at most 0.5 points on GSM8K at four loops from

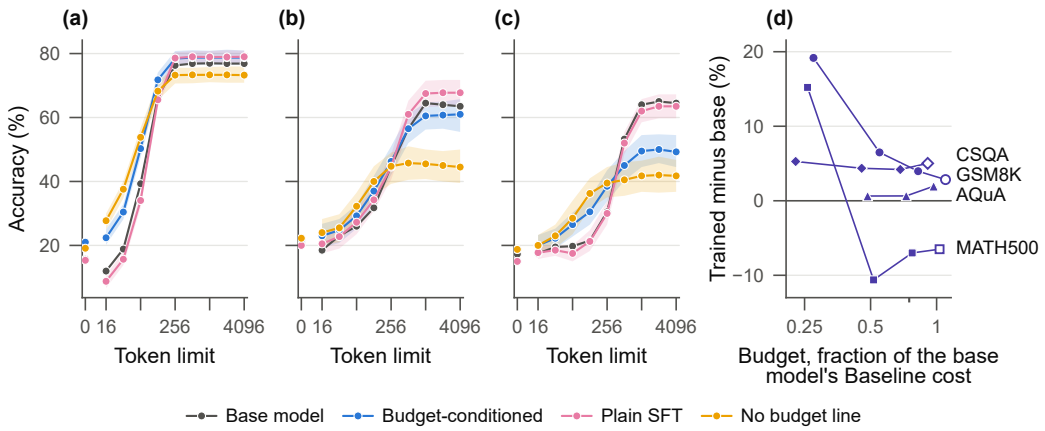

Figure 12: Ouro-1.4B after training. (a–c) Accuracy by token limit on GSM8K at four loops and MATH500 at four and two loops, with 95% confidence intervals for differences from the base model. (d) SETTLE-H accuracy difference between Budget-conditioned, uniform loops and the base model at matched generation-only costs, normalised by the base model's baseline cost. Hollow markers denote the trained variant's full-cost accuracy.

256 tokens onward, but by -7.2 points $[-12.0, -2.0]$ on MATH500 at two loops and 1024 tokens. Across six grids, absolute seed differences range from 0.6 to 3.9 points at 32 tokens and have a median of 2.3 points at 2048 tokens. Differences of this scale should be interpreted cautiously.

G Probing and Intervening on Internal Representations

Setup. We analyse Ouro-1.4B on 300 questions per task (154 for AQuA) at one to four loops. The budget-trained model is the Budget-conditioned, uniform loops variant (Appendix F). The lens applies the model's output head to intermediate states. Confidence intervals are 95% bootstrap intervals over questions.

Loop-index probes and steering. A linear probe identifies the loop index from answer-position states with 99.5% accuracy $[99.3, 99.7]$, versus 25% chance; probes across all 24 shared layers perform near perfectly on every task. On GSM8K at 128 tokens, adding the mean loop-4 minus loop-2 state difference after loop 2 raises the exit-gate logit from -5.35 to -0.24, close to the unmodified loop-4 value of -0.23. This moves the logit towards its loop-4 value on most questions, versus none for a norm-matched random direction, while changing intermediate and final answers less often.

Representations at settling. We measure KL divergence from the lens distribution at token limit T to the distribution at the model's natural stop, aligning questions by settle time J. Across four tasks at four loops, the budget-trained model approaches its final distribution at J, while the base model remains approximately 0.2 nats away; Plain SFT reaches 0.012 nats. Figure 5(b) shows GSM8K. Hidden-state differences remain nonzero, as the states follow different token prefixes.

Predicting settling and correctness. On GSM8K at 32 tokens, a linear settling probe achieves AUROC 0.83 $[0.78, 0.89]$, exceeding lens entropy, the truncation indicator, and the exit gate. It also performs best on MATH500 at 128 tokens. On GSM8K at 128 tokens, however, truncation alone achieves 0.91. Earlier settling predicts correctness across all four tasks (Figure 5(c)). At natural stopping, consecutive-loop KL divergence and the exit gate predict correctness with AUROCs of 0.55–0.65 and 0.58–0.60, respectively. An AQuA correctness probe reaches 0.81 $[0.73, 0.88]$ at loop 3, but steering along its direction at loop 4 corrects at most 9.0% of errors, versus 4.5% for a random direction.

Settling and commitment after training. On GSM8K at four loops, median settle time falls from 128 tokens for the base and Plain SFT models to 32 for the budget-trained model, which also has the largest settled share at 32 tokens (Figure 13(a)). We define commitment as the first token-limit and

Table 11: Accuracy difference from base Ouro-1.4B, in percentage points, on the same evaluation questions.

Variant	$T = 0$	$T = 128$	$T = 512$	$T = 4096$
GSM8K, four loops				
Budget-conditioned	$+5.1$	$+5.5$	$+2.1$	$+2.1$
Uniform sampling	$+4.6$	$+5.7$	$+2.6$	$+2.6$
Feasible budgets only	$+3.0$	$+4.0$	$+1.5$	$+1.5$
No budget instruction	$+3.7$	$+2.0$	-3.3	-3.4
Plain SFT	-0.3	-1.0	$+1.8$	$+1.8$
Plain SFT with mixed loops	-3.6	-1.5	$+1.7$	$+1.7$
MATH500, four loops				
Budget-conditioned	$+1.2$	$+4.8$	$+0.2$	-2.8
Uniform sampling	$+0.8$	$+0.0$	$+1.8$	$+3.4$
Feasible budgets only	$+1.2$	$+5.8$	$+0.4$	-1.4
No budget instruction	$+1.0$	$+7.0$	-11.4	-19.2
Plain SFT	-0.4	$+1.6$	$+2.6$	$+2.0$
Plain SFT with mixed loops	$+1.0$	-0.4	-1.8	$+2.4$
MATH500, two loops				
Budget-conditioned	$+1.8$	$+9.4$	-7.2	-14.0
Uniform sampling	$+0.6$	$+7.2$	-3.4	-6.4
Feasible budgets only	$+1.6$	$+8.8$	-8.4	-15.6
No budget instruction	$+1.4$	$+15.4$	-13.0	-22.8
Plain SFT	-2.4	$+0.2$	-0.4	$+0.4$
Plain SFT with mixed loops	$+0.2$	$+3.0$	-3.8	-7.6
AQuA, four loops				
Budget-conditioned	$+6.3$	$+0.4$	$+2.8$	$+2.8$
Uniform sampling	$+7.1$	-0.8	$+1.6$	$+2.0$
Feasible budgets only	$+7.1$	-2.4	$+0.0$	$+0.4$
No budget instruction	$+4.3$	-0.4	$+1.6$	$+1.6$
Plain SFT	$+0.4$	-0.8	$+2.4$	$+2.0$
Plain SFT with mixed loops	$+3.9$	-0.4	$+1.6$	$+1.2$
CommonsenseQA, four loops				
Budget-conditioned	$+4.9$	$+2.9$	$+2.9$	$+2.9$
Uniform sampling	$+4.6$	$+2.7$	$+2.7$	$+2.7$
Feasible budgets only	$+4.6$	$+2.2$	$+2.2$	$+2.2$
No budget instruction	$+4.0$	$+3.8$	$+3.8$	$+3.8$
Plain SFT	$+0.2$	$+1.6$	$+1.6$	$+1.6$
Plain SFT with mixed loops	$+2.6$	$+2.9$	$+2.9$	$+2.9$

Table 12: Budget-conditioned minus each control in points at each stated budget, mean over six grids of Ouro-1.4B (GSM8K and CSQA at four loops, AQuA at one and four loops, MATH500 at two and four loops), paired by question with 95% intervals from 400 resamples. A difference below the seed-to-seed floor of Appendix F.3 is not read as an effect.

Control	Stated budget (tokens)				
	0	32	128	512	2048
No budget instruction	$+0.8$ [-0.1, +1.7]	$+6.7$ [+5.2, +8.2]	$+1.0$ [-0.3, +2.1]	$+3.2$ [+1.9, +4.6]	$+3.9$ [+2.6, +5.4]
Feasible budgets only	$+0.9$ [+0.1, +1.8]	$+0.9$ [-0.2, +1.9]	$+0.6$ [-0.6, +1.7]	-0.0 [-1.2, +1.0]	-0.5 [-1.6, +0.6]
Uniform sampling	$+0.1$ [-0.8, +1.0]	$+0.7$ [-0.3, +1.8]	$+0.6$ [-0.8, +1.8]	-1.7 [-2.8, -0.5]	-2.8 [-4.0, -1.5]
Plain SFT	$+4.1$ [+3.0, +5.5]	$+16.2$ [+14.6, +17.7]	$+5.2$ [+3.7, +6.5]	-0.9 [-2.3, +0.5]	-2.3 [-3.5, -1.0]
Plain SFT mixed loops	$+3.9$ [+2.9, +5.1]	$+17.2$ [+15.6, +18.8]	$+4.7$ [+3.4, +6.0]	-0.3 [-1.6, +0.8]	-1.6 [-2.9, -0.2]

loop position where the lens top prediction matches the first token of the final answer. Most AQuA and CommonsenseQA questions reach commitment by 32 tokens in all three models. On MATH500, the budget-trained model commits before the base model's natural stopping point on 82% $[78, 86]$ of questions, more often than either comparison model. Its final accuracy is nevertheless lower: 53.7% versus 59.7% for the base model.

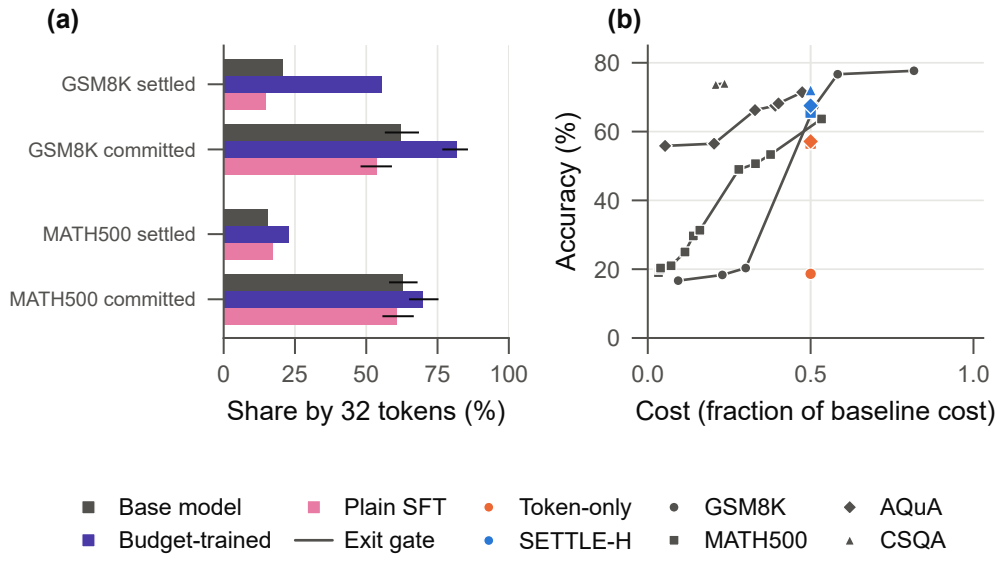

Figure 13: Ouro-1.4B. (a) Settled and committed shares at 32 tokens and four loops. (b) Best exit-gate accuracy at each generation-only cost, compared with Token-only and SETTLE-H at half the prefix-cached baseline cost. Error bars show 95% confidence intervals; the policies use different cost measures.

Adapter ablations. On GSM8K at 32 tokens, the budget-trained model scores 57.3%, versus 19.7% for the base model. Removing its adapter from layers 0–5 reduces accuracy by 10.3 points $[5.0, 15.7]$, the only removal with an interval excluding zero; later six-layer blocks have smaller effects. The same ablations change Plain SFT accuracy by at most 1.3 points. Among module types, removing down-projection adapters has the largest effect, with an interval of $[0.0, 7.0]$ points (Figure 5(d)). No removal has an interval excluding zero at 0 or 128 tokens, or on CommonsenseQA. The 55.0% result in Table 3 uses a different loop-sampling variant and all 1,219 GSM8K evaluation questions.

Weight updates. Budget-trained updates have norms equal to 1.0–2.4% of the corresponding module weights. Their cosine similarity to Plain SFT updates ranges from 0.17 to 0.29 across layers, with the six lowest values in layers 0–5.

Exit-gate comparison. Ouro's gate exits each token when its cumulative exit probability reaches a threshold. We evaluate thresholds $\{0.3, 0.5, 0.7, 0.9\}$ at the measured token limits. At 0.9, every token completes four loops; at 0.3, tokens use 2.2–3.3 loops on average. At half the baseline cost, the best gate setting matches four fixed loops at the same token limit on GSM8K, MATH500, and CommonsenseQA. On AQuA, it exceeds the affordable four-loop setting, limited to 32 tokens, by 14.9 percentage points (Figure 13(b)).