PROBECASCADE: VERIFICATION AND VALUE FOR EARLY EXIT IN LONG FORM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

In deployed long-form reasoning systems, models may continue generating substantial intermediate reasoning after a sufficiently reliable answer has emerged, directly increasing response latency and inference cost. Reasoning early stopping therefore requires recovering as much unnecessary computation as possible while keeping the accuracy loss introduced by early exits within an acceptable budget. Existing methods infer when to stop from local trajectory evidence such as answer stability, confidence, semantic redundancy, or internal representations, but strong local evidence can still coincide with an incorrect candidate and, even when reliability is high, does not by itself determine whether the current checkpoint is a valuable place to exit. We introduce ProbeCascade, a sentence-level early-stopping framework for accuracy-constrained computation recovery that explicitly separates checkpoint discovery from the decision to return the current candidate. ProbeCascade uses local convergence to trigger checkpoint inspection, estimates the reliability of the exposed candidate with a calibrated verifier, and combines this estimate with a Value signal that favors exits from which more computation can be recovered. It stops at the first eligible checkpoint whose score exceeds the selected threshold. To make efficiency gains reflect deployable savings rather than gross truncation, we charge every forced-answer probe actually issued by the stopping policy and report audited net token saving after monitoring overhead. We evaluate ProbeCascade on DeepSeek-R1-Distill-Qwen-7B with MATH-500, DeepSeek-R1-Distill-Llama-8B with MATH-500, and DeepSeek-R1-Distill-Qwen-7B with GSM8K under a common held-out evaluation protocol. On the primary Qwen-7B/MATH-500 setting, the policy selected under a nominal 1.0% accuracy-loss budget achieves 32.06% audited net token saving with 1.2 percentage points of observed accuracy loss, compared with 25.66% saving for Dynasor at the same observed-loss cap, while ProbeCascade also achieves positive audited computation recovery on both additional model–task settings.

1 INTRODUCTION

Long-form reasoning models solve difficult problems by continuing to generate intermediate reasoning steps (Wei et al., 2022; Wang et al., 2023; DeepSeek-AI, 2025). Yet trajectory length alone does not tell us when further reasoning is useful. Continuing after a reliable answer has emerged adds latency and decoding cost, while stopping too early can remove the reasoning that would have corrected a current error. Reasoning early stopping therefore asks when generation should end and the current candidate answer should be returned. A useful exit must return a sufficiently reliable answer while leaving enough remaining computation to make early exit worthwhile.

Existing reasoning early-stopping methods infer when to stop from evidence available along the current trajectory (Fu et al., 2025; Liu & Wang, 2025; Yang et al., 2026; Min et al., 2026; Dong et al., 2026). Such evidence includes repeated answers, confidence and entropy dynamics, semantic redundancy, output statistics, and internal representations. However, strong local evidence can occur when the current candidate is wrong. In our DeepSeek-R1-Distill-Qwen-7B traces on MATH-500, 343 of 500 trajectories contain at least one run of three identical wrong candidate answers followed by a correct answer later in the same trajectory. Confidence and entropy rank checkpoint correctness well overall, yet incorrect eligible checkpoints remain in the high-confidence and low-entropy tails. Figure 1 shows that repeated candidate agreement can precede later correction. Even at compara-

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.

ble verifier confidence, checkpoints can occur at different stages of trajectories and leave different amounts of reasoning ahead. A stable or highly certain candidate, a reliable answer, and a useful exit are therefore related but not equivalent.

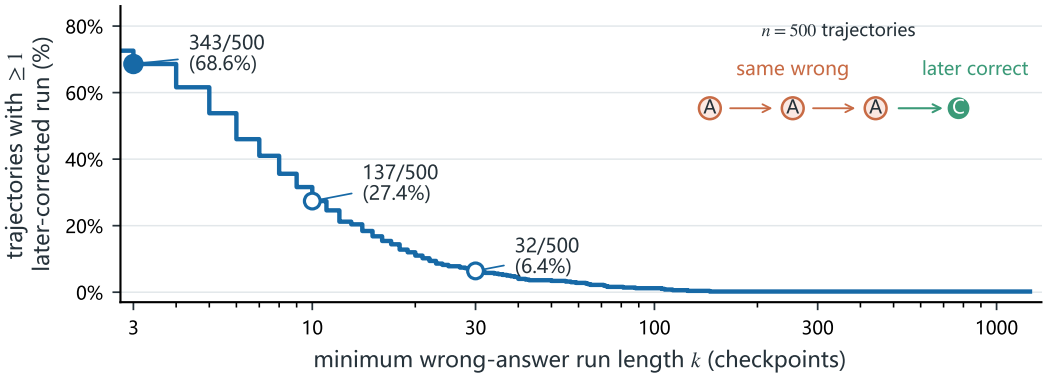

Figure 1: Wrong candidates can remain stable before later correction. The curve shows the fraction of MATH-500 trajectories containing at least one run of k identical incorrect candidates followed by a correct answer later in the same trajectory. At $k = 3$, this occurs in 343 of 500 trajectories (68.6%).

ProbeCascade is designed around this distinction. At each sentence checkpoint, a short forced-answer branch produces the candidate implied by the current reasoning prefix without altering the main reasoning branch. Recent candidate agreement and changes in confidence and entropy determine whether the checkpoint warrants a stopping decision. For such checkpoints, a calibrated verifier estimates the probability that returning the current candidate would be correct. Value is trained using the amount of reasoning that follows the checkpoint in completed training trajectories, so that the stopping rule favors exits from which more computation can be recovered. The verifier output and Value jointly determine the stopping score. When the stopping condition is met, ProbeCascade returns the current candidate. Otherwise, the original reasoning continues. Policy settings are selected under an accuracy-loss constraint, and every forced-answer probe already issued is included in the reported computation cost.

Our contributions are summarized as follows.

1. **Strong local evidence is not enough to stop.** We show that strong local evidence can still coincide with an incorrect candidate. The same incorrect candidate can recur across multiple checkpoints before later correction, and incorrect eligible checkpoints remain in high-confidence and low-entropy regions despite strong global ranking performance. These observations show why local trajectory evidence can identify checkpoints worth evaluating without being sufficient to determine whether the current answer should be returned.

2. **ProbeCascade decides when the current candidate should be returned.** We introduce ProbeCascade, which evaluates whether to return the current candidate using a calibrated estimate of answer correctness together with Value trained using the amount of reasoning that follows each checkpoint in completed training trajectories. The resulting stopping rule considers both whether the current candidate is reliable enough to return and whether stopping at that point can recover meaningful computation.

3. **Net savings include the cost of making stopping decisions.** We evaluate ProbeCascade on DeepSeek-R1-Distill-Qwen-7B with MATH-500, DeepSeek-R1-Distill-Llama-8B with MATH-500, and DeepSeek-R1-Distill-Qwen-7B with GSM8K. On the primary setting, ProbeCascade saves 32.1% of audited total generation cost with an observed accuracy reduction of 1.2 percentage points. We also examine how reducing probing frequency changes the realized computation–accuracy frontier while keeping the trained policy and selected parameters fixed.

2 Related Work

Long-form reasoning models need different amounts of computation for different problems. Difficult problems may continue to benefit from additional reasoning, whereas other problems reach a stable or reliable answer much earlier and do not require the same reasoning budget. Recent work has consequently moved from uniform test-time computation toward adaptive computation, where the amount of inference is adjusted according to problem difficulty or signals observed during reasoning. Snell et al. (2025) study how the effectiveness of test-time scaling depends on problem difficulty, while Certaindex/Dynasor (Fu et al., 2025) use the statistical stabilization of intermediate answers to allocate computation and terminate reasoning early. For a single long-form reasoning trajectory, existing early-stopping methods mainly rely on two kinds of online evidence: whether the reasoning process has become sufficiently stable, and whether the currently exposed candidate answer is sufficiently reliable.

One line of work derives stopping evidence from changes along the reasoning trajectory. Answer Convergence (Liu & Wang, 2025) studies when intermediate answers become consistent and develops both answer-consistency-based and learned stopping methods. CoDE-Stop (Hosseini et al., 2026) uses confidence dynamics of intermediate answers to determine whether further reasoning remains necessary. PUMA (Min et al., 2026) extends this view to reasoning states, using semantic redundancy between adjacent states to detect whether reasoning is still making meaningful progress while incorporating answer-level verification before exit. These methods can exploit information accumulated during generation and identify potential stopping locations before natural termination. Answer stability, confidence dynamics, and semantic redundancy, however, primarily characterize whether the reasoning trajectory is still changing; they are not equivalent to the correctness of the current candidate. A locally stable candidate may still be revised by subsequent reasoning. Trajectory evidence is therefore well suited to narrowing the region that warrants further inspection, but need not determine by itself whether the current answer should be returned.

Another line of work more directly evaluates whether the answer supported by the current reasoning prefix is sufficiently reliable. DEER (Yang et al., 2026) elicits trial answers at reasoning transitions and uses their confidence to decide whether to exit early. LearnStop (Dong et al., 2026) learns current-prefix correctness from prefix-observable features including confidence, entropy, vote share, answer stability, and backtracking, and further studies the effects of checkpoint schedules and probing overhead on early exit. Compared with reasoning-state stability alone, these approaches align the stopping decision more directly with the reliability of the current candidate and can better control premature exits. Reliability, however, does not uniquely determine the computational consequence of stopping. Two checkpoints with similar candidate reliability may occur at different stages of a trajectory and leave substantially different amounts of avoidable computation.

ProbeCascade formulates early stopping as a return decision that considers both candidate reliability and the computational value of stopping, rather than using trajectory stability or candidate reliability alone as the stopping criterion. Under an explicit accuracy-loss constraint, its stopping criterion considers whether the current candidate is sufficiently reliable and whether ending reasoning at the current checkpoint can yield meaningful net computation savings, while charging the actual probing cost to policy computation. Checkpoints with similar reliability are therefore not treated as equivalent stopping opportunities: an earlier reliable candidate with more avoidable computation remaining can receive a different stopping decision from one that appears near natural termination. Early stopping consequently depends not only on whether the current answer is reliable, but also on whether the current point still offers sufficient computational value to justify an exit.

Notation. We index problems by i and sentence-level checkpoints by $t \in \{1, \ldots, T_i\}$, and use F to denote natural full generation. The normalized candidate answer exposed at checkpoint t is $a_{i,t}$, the full-generation answer is $a_{i,F}$, and the gold answer is y_i. We write $a_{i,t} \equiv y_i$ for equivalence under the fixed task evaluator and $\mathbb{I}[\cdot]$ for the indicator function.

3 ProbeCascade

ProbeCascade makes stopping decisions at sentence boundaries, which we refer to as checkpoints. At each checkpoint, a side branch takes the current reasoning prefix, appends

</think>\n\boxed{, and greedily generates at most 64 tokens to expose the answer implied by the prefix. The generated box content is parsed as the candidate answer, while the original reasoning branch remains unchanged. Recent candidate agreement and uncertainty trends identify checkpoints worth evaluating for an early exit. At an eligible checkpoint, a verifier estimates the probability that returning the current candidate would be correct. Value provides a checkpoint-level signal trained from the unfinished suffixes of completed trajectories. The correctness estimate and Value form the stopping score used to either return the current candidate or continue reasoning. Figure 2 summarizes the procedure.

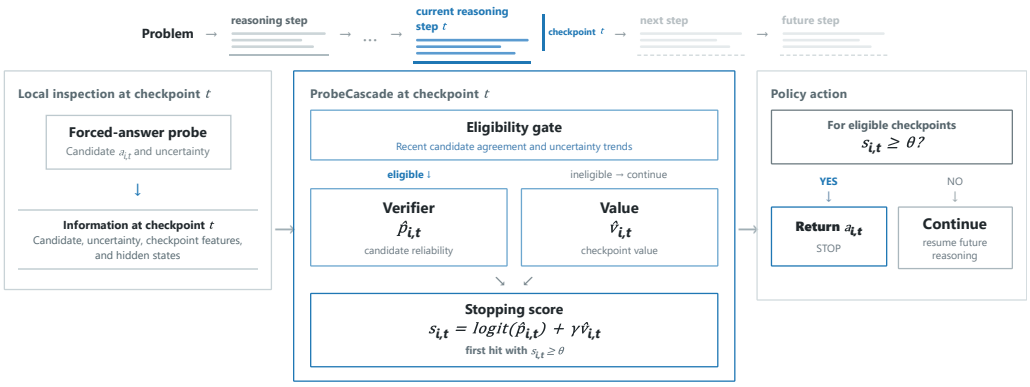

Figure 2: ProbeCascade at a sentence checkpoint. Recent candidate agreement and uncertainty trends determine whether the checkpoint is eligible for a stopping decision. At an eligible checkpoint, the verifier estimates the probability that returning the current candidate would be correct, while Value provides a second checkpoint-level signal trained from unfinished-suffix supervision. Their outputs form the stopping score. If the stopping condition is not met, reasoning continues to the next checkpoint.

3.1 Objective

Early stopping trades computation saving against answer accuracy. We seek the largest net token saving achievable under a prescribed accuracy-loss budget. For question i, let K_i^π denote the token cost incurred by policy π. For an early exit, this cost includes the main reasoning generated up to the stopping point and all forced-answer tokens generated by the policy. If the policy never exits early, it includes the full-generation cost together with all policy probes issued before natural termination. Let K_i^F denote the cost of the original full generation without stopping-policy probes. Corpus-level saving is

$$\mathrm{Save}(\pi) = 1 - \frac{\sum_i K_i^\pi}{\sum_i K_i^F}. \tag{1}$$

Using corpus-level token totals gives each trajectory weight in proportion to its actual computation. Let $\mathrm{Acc}(\pi)$ denote the accuracy under policy π, let $\mathrm{Acc}(F)$ denote full-generation accuracy, and let Π denote the candidate policy family. For an accuracy-loss budget ρ, we select

$$\pi_\rho^\star \in \arg\max_{\pi\in\Pi} \mathrm{Save}(\pi) \quad \text{s.t.} \quad \mathrm{Acc}(F) - \mathrm{Acc}(\pi) \leq \rho. \tag{2}$$

3.2 Eligibility

A sentence boundary is not automatically a useful point at which to consider stopping. Candidate answers can continue to change across early checkpoints, and their confidence and entropy can still vary substantially. ProbeCascade therefore uses recent checkpoint history to determine where the stopping rule is evaluated.

At checkpoint t, the forced-answer branch produces candidate answer $a_{i,t}$. Confidence and entropy are computed from the next-token vocabulary distribution immediately after the forced-answer pre-

fix. Confidence is the maximum token probability, while entropy is computed over the full vocabulary distribution. We compute least-squares slopes $\beta^c_{i,t}$ and $\beta^h_{i,t}$ over the four most recent observations of confidence and entropy. Candidate answers are normalized before agreement is evaluated. Eligibility is evaluated only after four observations are available. A checkpoint is eligible when the three most recent candidate answers agree and both uncertainty trends are sufficiently small.

$$E_{i,t} = \mathbb{I}[t \geq 4]\, \mathbb{I}[a_{i,t} = a_{i,t-1} = a_{i,t-2}]\, \mathbb{I}[|\beta^c_{i,t}| \leq \epsilon_c]\, \mathbb{I}[|\beta^h_{i,t}| \leq \epsilon_h]. \tag{3}$$

We use an agreement history of $m = 3$ in all experiments, while the confidence and entropy slopes use the four most recent observations. The three predefined eligibility settings differ only in (ϵ_c, ϵ_h). Eligibility determines which checkpoints are evaluated for stopping. It does not determine whether the current candidate is returned.

3.3 Answer Verification

Candidate agreement alone does not establish that the current answer is correct. We therefore train a verifier to estimate the probability that returning the current candidate at checkpoint t would be correct.

A checkpoint receives label $y_{i,t} = \mathbb{1}[a_{i,t} \equiv y_i]$, where y_i is the gold answer and $\equiv$ denotes the fixed answer-equivalence rule used throughout evaluation.

The verifier uses Compact32, a 32-dimensional feature vector consisting of three statistics from the next-token distribution at the start of the forced answer and 29 features derived from hidden states at eight uniformly sampled layers. Probe3 contains confidence, entropy, and the difference between the two largest token probabilities in this distribution. CrossLayer29 contains per-layer hidden-state changes between checkpoints, scaled hidden-state norms, adjacent-layer cosine similarities, and summary statistics across layers and checkpoints. Appendix E.2 gives the complete feature definition.

We fit an L_2-regularized logistic classifier with sample weights that balance questions and correctness labels. We then fit Platt scaling on out-of-fold classifier scores. The calibrated probability $\hat{p}_{i,t} = P(y_{i,t} = 1 \mid x_{i,t})$ estimates the correctness of $a_{i,t}$.

p

3.4 Value

Candidate correctness describes the risk of returning the current answer, but it is not the only information used by the stopping policy. We train a second checkpoint-level signal from the unfinished suffixes of completed training trajectories and call this signal Value.

Let L_i denote the length of the completed reasoning trajectory and $l_{i,t}$ the number of reasoning tokens generated by checkpoint t. We construct the supervision target

$$z_{i,t} = \log(1 + \max(L_i - l_{i,t}, 0))\,. \tag{4}$$

The complete trajectory is used only when constructing this target during training. At inference time, Value is computed only from information available at the current checkpoint.

The Value model takes five features: the current reasoning length, checkpoint index, recent answer stability, and the magnitudes of the confidence and entropy slopes. A Ridge regressor is fitted to $z_{i,t}$. We standardize its output using the mean and standard deviation estimated on the training data and denote the resulting value by $\hat{v}_{i,t}$.

Although the training target is constructed from the unfinished suffix, $\hat{v}_{i,t}$ is used only as a checkpoint-level signal in the stopping rule. We do not interpret its magnitude as a calibrated prediction of the number of tokens that remain.

3.5 Stopping Rule

If $E_{i,t} = 0$, the main reasoning branch continues. At an eligible checkpoint, ProbeCascade combines the calibrated correctness probability with Value:

$$s_{i,t} = \mathrm{logit}(\hat{p}_{i,t}) + \gamma \hat{v}_{i,t}. \tag{5}$$

Here, γ controls the contribution of Value. Setting $\gamma = 0$ removes Value and reduces the score to the verifier term.

For each accuracy-loss budget ρ, we jointly select the eligibility setting, γ, and stopping threshold θ on the fitting data according to Eq. 2. Evaluation then uses these selected parameters without further fitting. ProbeCascade processes checkpoints in chronological order and stops at the first eligible checkpoint satisfying $s_{i,t} \geq \theta$. It returns $a_{i,t}$ at that checkpoint. If no checkpoint satisfies the stopping condition, the model completes its original reasoning trajectory and returns the full-generation answer.

4 Experiments

4.1 Experimental Setup

Our primary evaluation uses DeepSeek-R1-Distill-Qwen-7B on MATH-500 (Hendrycks et al., 2021). We further evaluate DeepSeek-R1-Distill-Llama-8B on MATH-500 and DeepSeek-R1-Distill-Qwen-7B on GSM8K (Cobbe et al., 2021). Within each setting, all stopping methods are evaluated on the same complete reasoning trajectories.

Evaluation follows a five-fold question-level protocol, with all checkpoints from the same question assigned to the same fold. For each split, four folds are used to fit the learned components and select the stopping policy under the specified accuracy-loss budget, and the remaining fold is held out for evaluation. Across the five splits, this yields one held-out policy outcome for every question. We report accuracy loss relative to full generation and audited net token saving under the cost accounting in Section 3.1, including all probe tokens generated by the stopping policy.

Complete reasoning trajectories are generated with temperature 0.6, top-p 0.95, and a 32K-token context window. Sentence boundaries define checkpoints, following the procedure described in Section 3.

4.2 Baselines

We compare ProbeCascade with Dynasor, PUMA, and LearnStop, using Full Generation as the reference without early stopping. Dynasor uses Certaindex to measure statistical progress during reasoning and adapts inference computation accordingly. PUMA identifies candidate exit points from semantic redundancy across reasoning states and combines this signal with answer-level verification. LearnStop learns a stopping decision from prefix-observable statistics, including answer confidence, entropy, prefix vote share, answer stability, and backtracking-marker density. Because the original LearnStop operates on fixed-budget checkpoints, we retain its five input features and Logistic Regression correctness head while adapting the inspection points to our sentence-level checkpoint protocol; policy selection and cost accounting follow the common evaluation protocol used throughout this work.

4.3 Main Results

Table 1: Audited net token saving (%) on Qwen-7B/MATH-500 under matched observed accuracy-loss caps. The ProbeCascade entry at 1.2 pp corresponds to the policy selected under the nominal 1.0% accuracy-loss budget.

Loss cap (pp)	ProbeCascade	Dynasor	PUMA	LearnStop (adapted)
0.6	19.43	12.42	0.00	0.00
0.8	19.43	24.01	0.00	0.00
1.0	22.46	24.01	0.00	0.00
1.2	32.06	25.66	0.00	0.00
2.0	32.06	25.66	0.00	0.00
3.0	37.19	34.77	0.00	0.00
5.0	41.68	40.80	29.15	12.99

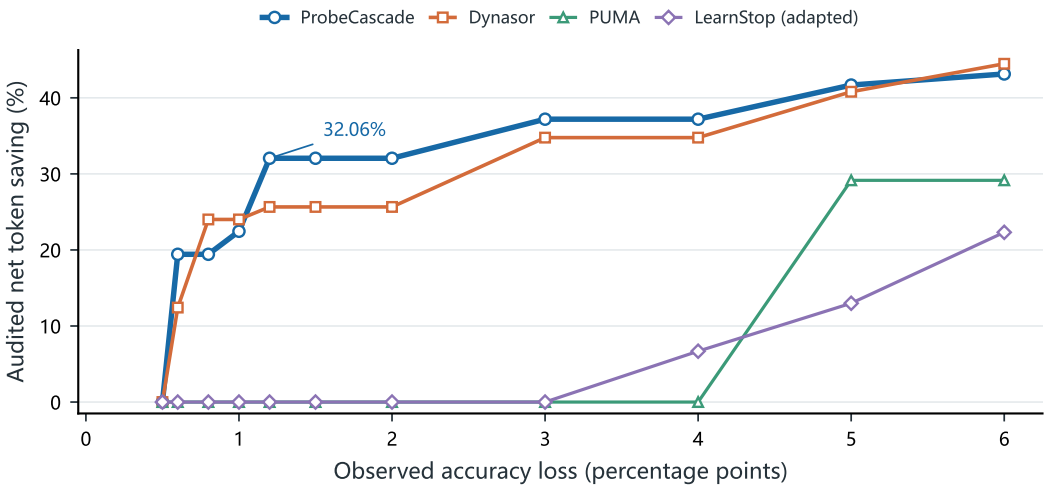

Figure 3: Accuracy–computation frontier on DeepSeek-R1-Distill-Qwen-7B with MATH-500. Audited net token saving includes all forced-answer probe costs. Points show measured held-out operating points under observed accuracy loss.

Figure 3 and Table 1 show the accuracy–computation frontier on Qwen-7B/MATH-500. Under the nominal 1.0% accuracy-loss budget, ProbeCascade achieves 32.06% audited net token saving, with an observed accuracy loss of 1.2 percentage points on held-out questions. At the same 1.2 pp observed-loss cap, Dynasor achieves 25.66% audited saving, leaving a 6.40 percentage-point gap in net computation recovery.

As the allowed accuracy loss increases, ProbeCascade recovers more computation, reaching 37.19% and 41.68% audited saving at the 3.0 pp and 5.0 pp caps, respectively. PUMA and the adapted LearnStop baseline begin to yield substantial net savings at more permissive risk levels, while ProbeCascade already recovers considerable computation in the low-risk regime emphasized in this work.

4.4 Results Across Models and Tasks

We further evaluate ProbeCascade on DeepSeek-R1-Distill-Llama-8B with MATH-500 and DeepSeek-R1-Distill-Qwen-7B with GSM8K. Table 2 reports audited net token saving together with the observed accuracy loss of the selected operating point.

Table 2: ProbeCascade across additional model and task settings. Each entry reports audited net token saving / observed accuracy loss (%).

Loss cap (pp)	Llama-8B / MATH-500	Qwen-7B / GSM8K
1	26.58 / 1.00	10.76 / 0.15
5	39.31 / 4.80	21.84 / 2.27
10	47.64 / 9.80	21.84 / 2.27

ProbeCascade achieves positive audited computation recovery in both additional settings. At the 1 pp cap, it saves 26.58% on Llama-8B/MATH-500 and 10.76% on Qwen-7B/GSM8K, with observed losses of 1.00 and 0.15 pp, respectively. Saving increases at more permissive caps, as shown in Table 2.

4.5 Value and Monitoring Cost

Figure 4(a) isolates the contribution of Value by removing the Value term while keeping the remaining stopping pipeline fixed. Adding Value increases audited net token saving by 22.46, 13.85,

and 18.99 percentage points at the 1, 2, and 3 pp observed accuracy-loss caps, respectively. Value therefore expands the attainable accuracy–computation frontier under the same stopping protocol.

Monitoring also consumes computation through forced-answer probes. We evaluate this cost by replaying the same fitted policy while inspecting every second or every third checkpoint, without retraining or reselecting policy parameters. Candidate histories are recomputed causally from the probes actually issued under each schedule. As shown in Figure 4(b), Every-2 reduces probe calls and probe tokens to approximately 51% of the Every-1 level, while Every-3 reduces them to approximately 36%. At the 1 pp observed-loss cap, audited saving increases from 22.46% with Every-1 to 30.65% with Every-2 and 39.27% with Every-3. At the 5 pp cap, the corresponding savings are 41.68%, 47.13%, and 45.83%. Substantially fewer probes can therefore retain strong audited computation recovery while reducing monitoring cost.

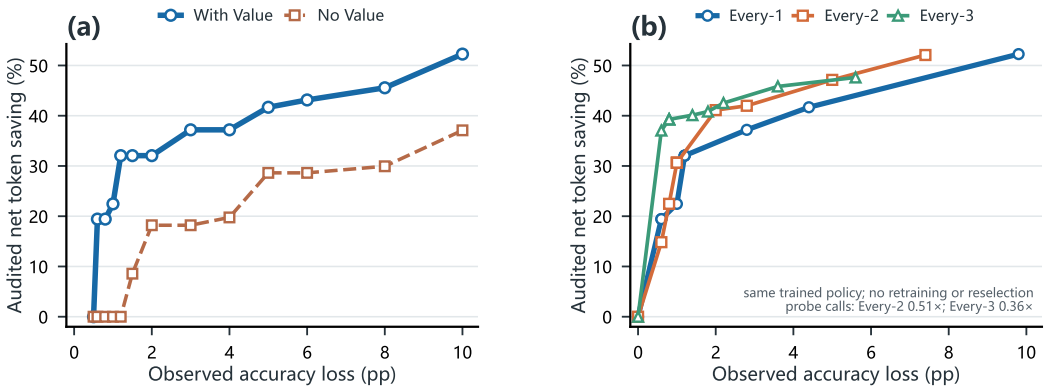

Figure 4: Value and monitoring-cost analysis on Qwen-7B/MATH-500. (a) Audited accuracy–computation frontiers with and without Value while keeping the remaining stopping pipeline fixed. (b) Frozen-policy replay with Every-1, Every-2, and Every-3 checkpoint inspection. Sparse schedules use only the probes actually issued and do not retrain or reselect policy parameters.

4.6 Exit Outcomes and Case Studies

Beyond aggregate accuracy loss, we further examine how early exit changes the returned outcome relative to full generation. Under the policy selected with the nominal 1.0% accuracy-loss budget, ProbeCascade exits early on 373 of 500 trajectories. Among these exits, 337 preserve a correct full-generation outcome, 11 replace a correct full-generation answer with an incorrect early return, and 5 exhibit the opposite transition, preserving a correct intermediate answer that full reasoning later changes to an error. Another 20 exits remain incorrect under both early stopping and full generation. The resulting held-out accuracy loss therefore reflects the net effect of harmful and beneficial answer transitions rather than premature stopping in only one direction. As the operating point becomes more permissive, the number of correct-preserving exits quickly saturates, while harmful transitions increase from 11 to 19 and 27 and beneficial transitions remain at 5. Additional computation recovery at looser operating points is therefore accompanied by increasingly risky exits. The complete outcome decomposition is reported in the appendix.

Figure 5(a) shows why local agreement alone cannot authorize an exit. The model repeatedly produces the same incorrect answer, 60°, at checkpoints 9–11, but ProbeCascade does not accept the candidate at checkpoint 11 ($s = -1.02 < \theta = 0.89$). Reasoning therefore continues, the answer changes to 30° at checkpoint 15, and the trajectory is eventually returned correctly. The example illustrates that repeated agreement can identify a checkpoint worth inspecting without implying that the current candidate is ready to return.

Figure 5(b) illustrates a different stopping behavior. At checkpoint 41, the Full-Value and No-Value policies encounter the same correct candidate, 8, but produce substantially different first-hit exit times. Full Value returns at checkpoint 41, whereas No Value continues to checkpoint 180 before returning the same answer. The earlier exit reduces audited cost by 4,065 tokens while preserving

the same correctness outcome, showing that checkpoint value can alter first-hit exit timing even when both policies ultimately return a correct answer.

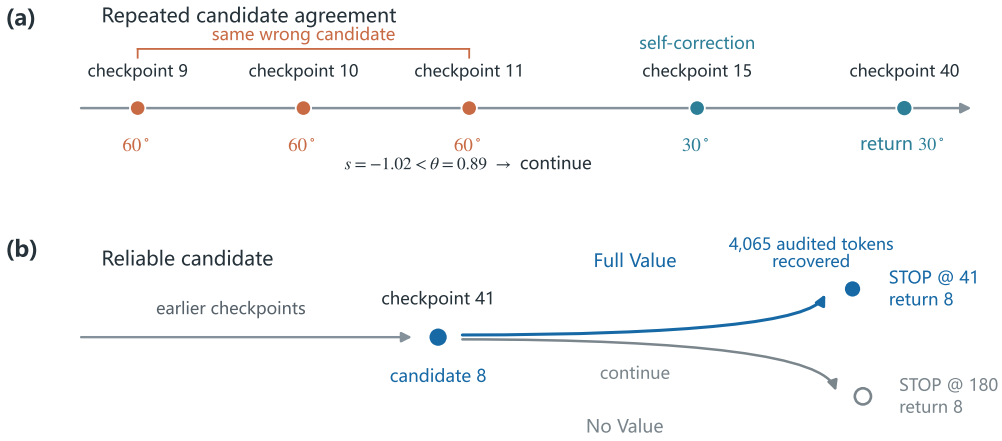

Figure 5: Held-out trajectories illustrating two stopping behaviors. (a) A locally stable but incorrect candidate is not accepted, preserving the continuation that later self-corrects from 60° to 30°. (b) On the same correct candidate, the Full-Value policy exits at checkpoint 41 while the No-Value policy continues to checkpoint 180; the earlier exit reduces audited cost by 4,065 tokens.

5 Conclusion and Future Work

This work studies how to identify checkpoints at which early exit is safe after local convergence emerges in long-form reasoning, while terminating reasoning that is no longer necessary as early as possible. Our analysis shows that local answer stability alone does not guarantee safe stopping. On Qwen-7B/MATH-500, 68.6% of complete trajectories contain three consecutive identical incorrect candidates that are later corrected by subsequent reasoning. Based on this observation, we propose ProbeCascade, which uses local convergence as a trigger for checkpoint inspection, employs a verifier to assess the reliability of the current candidate answer, and further evaluates checkpoint value to improve first-hit exit timing.

Experimental results show that ProbeCascade achieves an effective trade-off between accuracy loss and computation recovery. On DeepSeek-R1-Distill-Qwen-7B/MATH-500, the policy selected under the nominal 1.0% accuracy-loss budget achieves 32.06% audited net token saving with a held-out accuracy loss of 1.2 pp. This result accounts for the cost of all forced-answer probes actually issued. ProbeCascade also achieves positive audited computation recovery on Llama-8B/MATH-500 and Qwen-7B/GSM8K. Further analysis shows that reducing checkpoint inspection frequency can substantially reduce probe usage while retaining strong net computation recovery. Overall, the results show that ProbeCascade can recover substantial reasoning computation while controlling accuracy loss, and that these gains remain after monitoring cost is accounted for.

The current results also naturally point to two further directions. First, probe scheduling can be extended from a fixed frequency to an adaptive decision. Sparse replay has shown substantial redundancy in dense monitoring, motivating further study of issuing a forced-answer probe only when the current trajectory state is likely to change the stopping decision. Second, the likelihood of answer revision after the current prefix can be modeled explicitly. ProbeCascade currently decides whether to exit based on prefix-observable information, while some trajectories may still change their answers during subsequent reasoning even after the current candidate has exhibited high reliability. If such revision risk can be estimated from the current prefix, it can complement candidate verification and Value to better distinguish checkpoints that are already ready to return from those that still benefit from continued reasoning.

AI USE STATEMENT

We used generative AI tools to aid and polish writing, retrieve and discover related work, and assist with research ideation and execution, including methodology discussion, implementation and debugging, analysis of experimental results, and figure preparation. All AI-assisted work was reviewed by the authors, who take full responsibility for the final content.

REFERENCES

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.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Zhe Dong, Fang Qin, and Manish Shah. When does learning to stop help? a cost-aware study of early exits in reasoning models. *arXiv preprint arXiv:2606.30852*, 2026. doi: 10.48550/arXiv.2606.30852. URL https://arxiv.org/abs/2606.30852.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling LLM reasoning programs with certaindex. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4751. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d037fd021c9aace128b8ce25001cdb6c-Abstract-Conference.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Parsa Hosseini, Sumit Nawathe, Mahdi Salmani, Meisam Razaviyayn, and Soheil Feizi. Early stopping for large reasoning models via confidence dynamics. *arXiv preprint arXiv:2604.04930*, 2026. doi: 10.48550/arXiv.2604.04930. URL https://arxiv.org/abs/2604.04930.

Xin Liu and Lu Wang. Answer convergence as a signal for early stopping in reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17896–17907. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.904. URL https://aclanthology.org/2025.emnlp-main.904/.

Dehai Min, Giovanni Vaccarino, Huiyi Chen, Yongliang Wu, Gal Yona, and Lu Cheng. Stop when reasoning converges: Semantic-preserving early exit for reasoning models. *arXiv preprint arXiv:2605.17672*, 2026. doi: 10.48550/arXiv.2605.17672. URL https://arxiv.org/abs/2605.17672.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

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. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8df90a1440ce782d1f5607b7a38f2531-Abstract-Conference.html.

APPENDIX / SUPPLEMENTARY MATERIAL

A OUTCOMES OF EARLY EXITS

Figure 6 extends the analysis in Section 4.6 to three operating points with different observed accuracy losses.

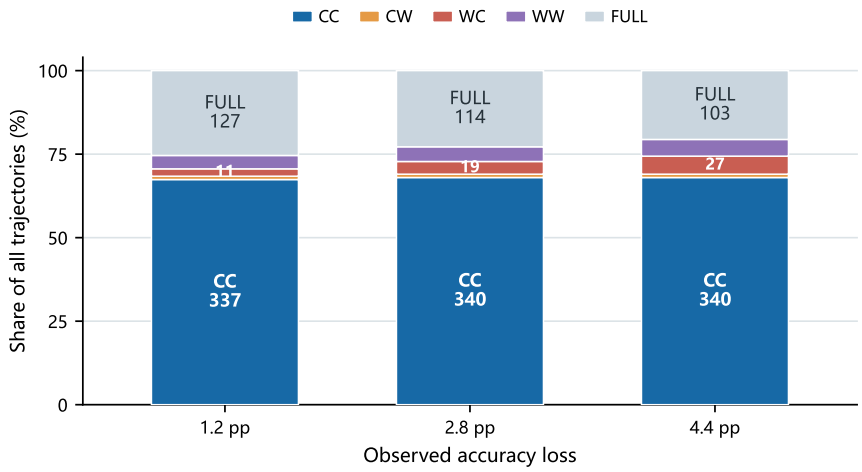

Figure 6: Distribution of outcomes across three operating points on Qwen-7B/MATH-500. For early exits, the first letter indicates whether the returned answer is correct (C) or wrong (W), and the second indicates the outcome of full generation. FULL denotes questions for which the policy does not stop early. Each bar represents all 500 questions.

The number of early exits for which both answers are correct changes little across the three operating points, increasing from 337 to 340. In contrast, the number of exits that return a wrong answer when full generation is correct increases from 11 to 27. These results show how the distribution of outcomes changes as the policy accepts greater accuracy loss.

B VERIFIER ANALYSIS

We examine the ability of the available signals to distinguish correct from incorrect candidates at eligible checkpoints. On Qwen-7B/MATH-500, confidence, negative entropy, and the probability margin achieve AUROC values of 0.9383, 0.9387, and 0.9374, respectively. Despite their strong discrimination, 811 incorrect eligible checkpoints have confidence above 0.99, and 1,168 have entropy below 0.10. These observations illustrate why local evidence is informative but does not establish the correctness of an individual candidate.

We further compare three feature representations within the stopping policy: Probe3, which uses statistics from the next-token distribution at the start of the forced answer; CrossLayer29, which uses hidden-state features; and Compact32, which combines both. Figure 7 presents their accuracy and computation trade-offs under a common evaluation protocol for this comparison.

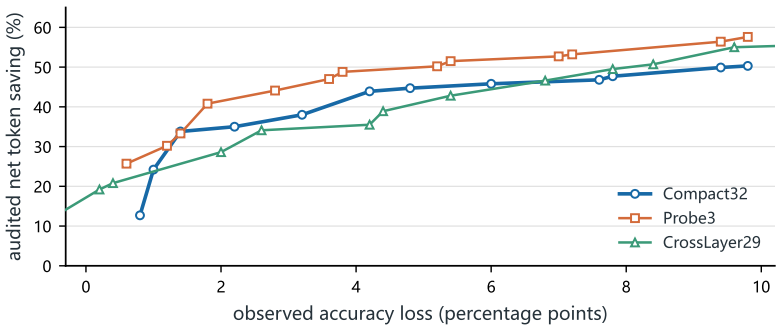

Figure 7: Audited net token saving against observed accuracy loss on Qwen-7B/MATH-500 for three verifier feature representations.Probe3 uses statistics from the next-token distribution at the start of the forced answer, CrossLayer29 uses hidden-state features, and Compact32 combines both representations.

Table 3: Audited net token saving for three Value variants on Qwen-7B/MATH-500. Each row represents an observed accuracy-loss cap. Saving values are percentages, and the final column reports the difference in percentage points between Full Value and No Value.

Loss cap (pp)	No Value	Token-step	Full Value	Full − No Value
0.5	0.00	0.00	0.00	0.00
0.6	0.00	19.12	19.43	19.43
0.8	−0.01	21.47	19.43	19.43
1.0	−0.01	21.47	22.46	22.46
1.2	−0.01	24.32	32.06	32.06
1.5	8.57	24.32	32.06	23.49
2.0	18.20	24.32	32.06	13.85
3.0	18.20	35.92	37.19	18.99
4.0	19.76	36.78	37.19	17.43
5.0	28.63	41.36	41.68	13.05
6.0	28.63	42.41	43.14	14.51
8.0	29.94	44.06	45.56	15.62
10.0	37.09	50.83	52.26	15.17

C Value Contribution

C.1 Contribution to Computation Recovery

We examine the contribution of Value by comparing the full method with No Value and Token-step, a control based on reasoning progress. Table 3 reports the audited net token saving of all three variants across the measured range of accuracy-loss constraints.

Full Value achieves higher audited saving than No Value at every measured cap above 0.5 pp. Token-step also recovers substantial computation, showing that reasoning progress provides a strong control. The difference between Full Value and Token-step varies across operating points: Full Value provides an additional 7.74 percentage points of saving at the 2 pp cap, whereas Token-step achieves 2.04 percentage points more at the 0.8 pp cap. The complete results therefore establish the contribution of Value while showing how much of the improvement is attainable using a simple measure of progress.

We further compare Full Value and No Value using paired bootstrap resampling. Table 4 reports the differences at three representative accuracy-loss caps.

Table 4: Paired comparison of Full Value and No Value on Qwen-7B/MATH-500. Differences are calculated as Full Value minus No Value. The 95% confidence intervals for audited saving are obtained from 15,000 paired bootstrap resamples of questions.

Loss cap (pp)	Saving difference (pp)	95% CI (pp)	Accuracy difference (pp)
1.0	22.46	[6.71, 42.15]	−0.20
2.0	13.85	[1.36, 31.87]	0.80
3.0	18.99	[6.33, 37.08]	−0.80

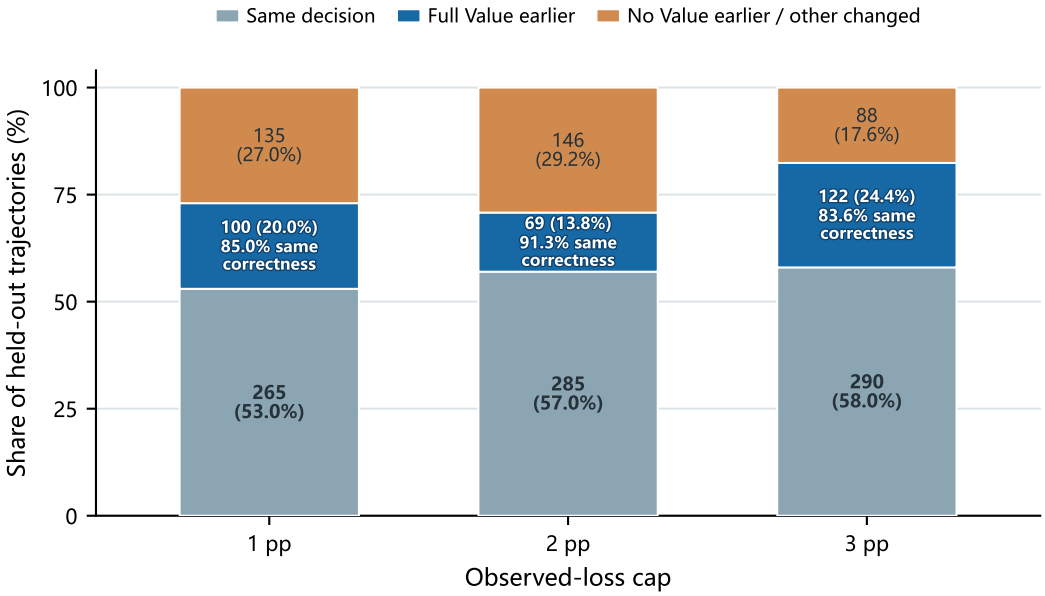

Figure 8: Comparison of stopping decisions with and without Value on Qwen-7B/MATH-500. Each bar represents all 500 evaluation questions. The categories distinguish identical stopping decisions, questions on which Full Value stops earlier, and other changes in the selected decision. The annotations for earlier exits indicate the percentage of questions on which both policies achieve the same correctness outcome.

The confidence intervals for the difference in audited saving remain above zero at all three operating points. The corresponding accuracy differences vary in direction, while Full Value consistently achieves greater computation recovery in these comparisons.

C.2 Changes in Stopping Decisions

To examine how Value affects the deployed policy, we compare the stopping decisions selected by Full Value and No Value on the same evaluation questions. Figure 8 summarizes these comparisons at three observed accuracy-loss caps.

Value changes the selected stopping decision on 47%, 43%, and 42% of questions at the 1, 2, and 3 pp caps, respectively. Full Value stops earlier on 100, 69, and 122 questions at these operating points. Among these earlier exits, the two policies achieve the same correctness outcome in 85.0%, 91.3%, and 83.6% of cases, respectively. The example in Section 4.6 illustrates how Value can select a substantially earlier exit while retaining the correct answer.

Table 5: Audited net token saving (%) for three monitoring schedules on Qwen-7B/MATH-500. Each column reports the results under the specified observed accuracy-loss caps. The fitted policies are replayed without retraining or parameter reselection.

Loss cap (pp)	Every-1	Every-2	Every-3
0.5	0.00	0.00	0.00
0.6	19.43	14.83	37.05
0.8	19.43	22.47	39.27
1.0	22.46	30.65	39.27
1.2	32.06	30.65	39.27
1.5	32.06	30.65	40.11
2.0	32.06	41.11	40.83
3.0	37.19	41.98	42.52
5.0	41.68	47.13	45.83
10.0	52.26	52.08	47.61

Table 6: Changes in audited saving relative to Every-1 on Qwen-7B/MATH-500. The two cost components sum to the net change, subject to rounding. Positive values indicate additional saving.

Schedule	Loss cap (pp)	Net change (pp)	Probe component (pp)	Generation component (pp)
Every-2	1.0	8.19	10.12	−1.93
Every-2	5.0	5.45	6.42	−0.97
Every-2	10.0	−0.18	3.71	−3.89
Every-3	1.0	16.82	15.45	1.36
Every-3	5.0	4.14	7.99	−3.84
Every-3	10.0	−4.65	5.05	−9.70

D Additional Experimental Analysis

D.1 Sparse Monitoring

We examine the effect of reducing the frequency of forced-answer probes. Every-2 and Every-3 inspect every second and third checkpoint, respectively. Both schedules replay the fitted policies without retraining or parameter reselection, and candidate histories are constructed using only the probes issued under each schedule. Table 5 reports the complete results.

Reducing the number of probes does not necessarily reduce the total computation cost. Sparse monitoring saves probe tokens but can also change the checkpoints at which the policy stops. To distinguish these effects, Table 6 separates the change in probe cost from the change in main generation cost, using Every-1 as the reference.

At the 1 and 5 pp caps, both sparse schedules achieve greater net saving than Every-1, primarily through lower probe costs. At the 10 pp cap, the additional main generation cost offsets the reduction in probe tokens. The results illustrate the trade-off between monitoring overhead and the opportunity to stop at earlier checkpoints.

D.2 Sensitivity Analysis

We examine the sensitivity of the stopping policy to candidate agreement history, the contribution of Value, and the uncertainty thresholds used by the eligibility test. Figure 9 shows the resulting accuracy and computation trade-offs. The main experiments fix $m = 3$ and jointly select the predefined eligibility setting, γ, and θ using the fitting folds.

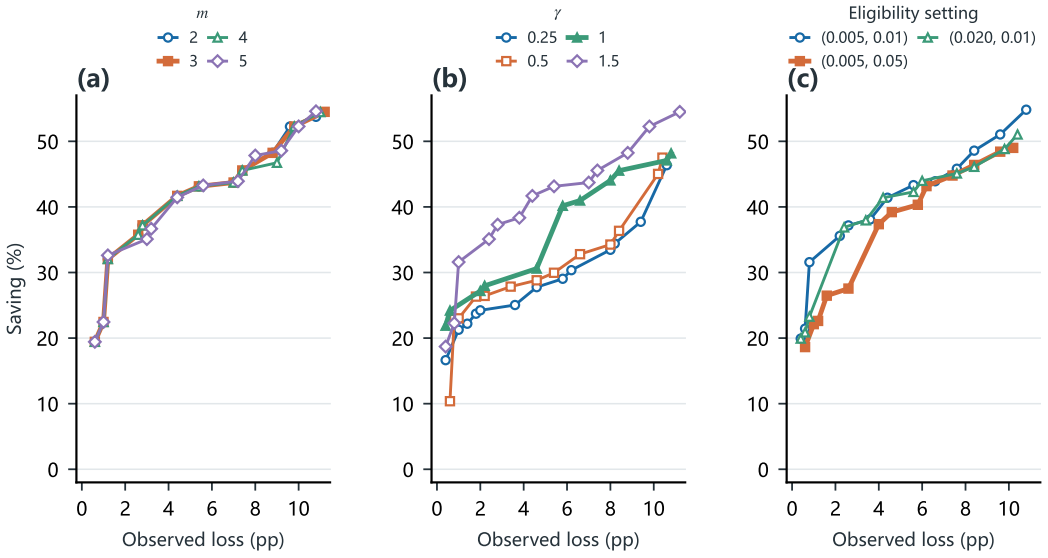

Figure 9: Sensitivity of audited net token saving to policy settings on Qwen-7B/MATH-500. The panels compare (a) the length m of the candidate agreement history, (b) the weight γ assigned to Value, and (c) the confidence and entropy slope thresholds used by eligibility rule.

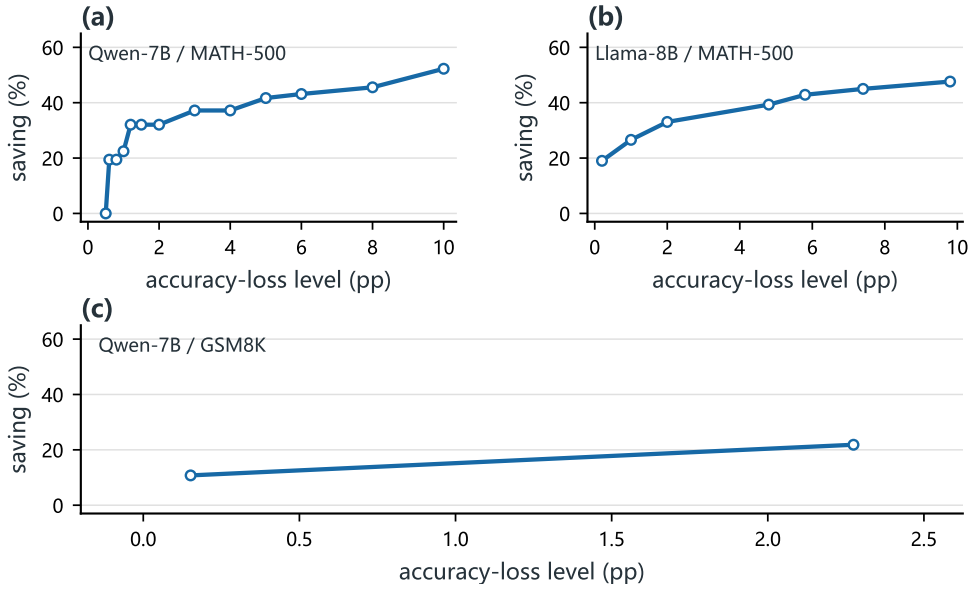

Figure 10: Audited net token saving across observed accuracy-loss levels for ProbeCascade on (a) Qwen-7B/MATH-500, (b) Llama-8B/MATH-500, and (c) Qwen-7B/GSM8K.

D.3 Complete Experimental Results

Figure 10 shows the measured trade-offs across the three evaluation settings.

Table 7 presents the complete results of the primary comparison across all reported accuracy loss limits, including the results for each baseline. Table 8 extends the evaluation to Llama-8B/MATH-500 and Qwen-7B/GSM8K, reporting both audited net token saving and observed accuracy loss at each operating point.

Table 7: Complete measured accuracy and computation trade-offs on Qwen-7B/MATH-500. Entries report audited net token saving (%) under the specified observed accuracy-loss caps.

Loss cap (pp)	ProbeCascade	Dynasor	PUMA	LearnStop (adapted)
0.5	0.00	0.00	0.00	0.00
0.6	19.43	12.42	0.00	0.00
0.8	19.43	24.01	0.00	0.00
1.0	22.46	24.01	0.00	0.00
1.2	32.06	25.66	0.00	0.00
1.5	32.06	25.66	0.00	0.00
2.0	32.06	25.66	0.00	0.00
3.0	37.19	34.77	0.00	0.00
4.0	37.19	34.77	0.00	6.69
5.0	41.68	40.80	29.15	12.99
6.0	43.14	44.48	29.15	22.32
8.0	45.56	47.30	49.68	30.02
10.0	52.26	50.73	53.68	31.75

Table 8: Additional operating points for Llama-8B/MATH-500 and Qwen-7B/GSM8K. Saving is reported as a percentage of the full generation cost, and observed accuracy loss is reported in percentage points.

Setting	Loss cap (pp)	Saving (%)	Observed loss (pp)
Llama-8B / MATH-500	0.5	19.00	0.20
Llama-8B / MATH-500	1.0	26.58	1.00
Llama-8B / MATH-500	2.0	33.05	2.00
Llama-8B / MATH-500	3.0	33.05	2.00
Llama-8B / MATH-500	5.0	39.31	4.80
Llama-8B / MATH-500	6.0	42.86	5.80
Llama-8B / MATH-500	8.0	44.99	7.40
Llama-8B / MATH-500	10.0	47.64	9.80
Qwen-7B / GSM8K	0.5	10.76	0.15
Qwen-7B / GSM8K	1.0	10.76	0.15
Qwen-7B / GSM8K	2.0	10.76	0.15
Qwen-7B / GSM8K	3.0	21.84	2.27
Qwen-7B / GSM8K	5.0	21.84	2.27
Qwen-7B / GSM8K	6.0	21.84	2.27
Qwen-7B / GSM8K	8.0	21.84	2.27
Qwen-7B / GSM8K	10.0	21.84	2.27

For PUMA, the zero entries at the stricter caps indicate that no stopping policy achieves positive audited saving within those constraints. LearnStop (adapted) has feasible stopping policies at some of these caps, but their audited saving is negative, so the reported results retain the option of continuing full generation.

E Implementation and Evaluation Details

E.1 Generation and Checkpoints

The evaluation uses the three model and benchmark settings summarized in Table 9. Stopping decisions are evaluated at sentence boundaries in the recorded reasoning trajectories. Each inspected checkpoint invokes a forced-answer probe, which terminates when the answer box is completed or

Table 9: Evaluation datasets and recorded sentence checkpoints. Full accuracy is measured using the fixed answer evaluator.

Setting	Questions	Checkpoints	Full acc. (%)
Qwen-7B / MATH-500	500	120,669	93.0
Llama-8B / MATH-500	500	87,423	77.6
Qwen-7B / GSM8K	1,319	57,182	88.9

the generation limit is reached. The probe produces a candidate answer without modifying the original reasoning trajectory.

For candidate agreement, the probe answer is taken from the recorded candidate field when available, with the generated `\boxed{...}` content used as a fallback. Candidate answers are normalized by removing commas and dollar signs; numeric answers are converted to a canonical floating-point representation, while non-numeric answers are lowercased with whitespace removed. This normalization is used only to compare candidates across checkpoints. Correctness is determined separately by the fixed task evaluator.

The stopping policy uses the sentence checkpoint grid. Candidate histories are constructed from the probes actually issued by the monitoring schedule. In particular, the agreement and uncertainty statistics under sparse monitoring are computed from the most recent observed probes rather than from checkpoints that were skipped.

E.2 Feature Definitions and Training

The verifier uses Compact32, which combines three statistics from the next-token distribution at the start of the forced answer with 29 features computed from hidden states at eight selected layers. The hidden-state features describe both the current representation and its change from the preceding checkpoint. The selected layers are 4, 7, 10, 14, 17, 21, 24, and 28. Table 10 specifies the complete feature representation.

The verifier is trained using weighted L2-regularized logistic regression with $C = 1.0$. Sample weights balance individual questions and the two correctness classes. Within each outer training split, classifier predictions are obtained from additional folds constructed at the question level. An independent Platt model is fitted to these predictions and applied to the scores produced by the classifier trained on the full outer training split. Feature standardization and calibration are fitted without using the held-out questions.

Value is trained using five features available during inference: $\log(1 + \ell_{i,t})$, $\log(2 + t)$, recent candidate stability, and the absolute confidence and entropy slopes. The training target is $z_{i,t} = \log(1 + \max(L_i - \ell_{i,t}, 0))$, where L_i is the length of the completed training trajectory and $\ell_{i,t}$ denotes the reasoning length at checkpoint t. We use Ridge regression with $\alpha = 10$ and standardize its predictions using the mean and standard deviation of predictions obtained on the outer training data through additional folds.

Token-step uses the same training target and Ridge regression procedure but retains only the two progress features, $\log(1 + \ell_{i,t})$ and $\log(2 + t)$. No Value sets $\gamma = 0$ in the stopping score. The three variants are evaluated using the same data partitions, policy selection procedure, and computation accounting.

E.3 Policy Selection and Evaluation

Evaluation follows five fixed folds constructed at the question level. For the primary setting, each fold contains 400 training questions and 100 held-out questions. The verifier and Value are fitted using only the corresponding training questions. For each nominal accuracy-loss budget, the predefined eligibility setting, γ, and θ are jointly selected on the training data. The selected policy is then applied to the held-out questions without further parameter selection.

The nominal budget constrains policy selection using the training questions. The observed accuracy loss is measured independently on the held-out questions and may differ from the nominal budget.

Table 10: Compact32 feature specification. Positions are numbered from 1 to 32. Features involving the preceding checkpoint use the recorded representation from that checkpoint.

Positions	Feature specification
1–3	Candidate confidence, entropy, and the probability margin between the two most likely tokens.
4–11	Euclidean norms of the differences between current and previous checkpoint hidden states, computed separately at each selected layer.
12–19	Hidden-state norms at the current checkpoint, divided by 100.
20–26	Cosine similarities between adjacent selected layers at the current checkpoint.
27–29	Mean, standard deviation, and ratio of the largest to the smallest hidden-state norm.
30	Mean cosine similarity between current and preceding checkpoint representations.
31	Index of the layer with the largest representation change, normalized by eight.
32	Mean cosine similarity over all pairs of selected layers.

Table 11: Parameter selection for the main stopping policy. The eligibility setting, γ, and θ are selected jointly for each outer fold and nominal budget.

Parameter	Candidate values or procedure
eligibility thresholds (ϵ_c, ϵ_h)	$(0.005, 0.05)$, $(0.005, 0.01)$, $(0.020, 0.01)$
Agreement history m	3 for the main experiments
Value weight γ	0.25, 0.5, 1.0, 1.5
Stopping threshold θ	Selected from the event frontier on the training questions
Selection criterion	Maximum audited saving subject to the nominal accuracy-loss budget

Table 12 reports the selected parameters and evaluation results for the nominal 1.0% budget on Qwen-7B/MATH-500.

Across 500 held-out questions, the selected policy achieves 91.8% accuracy and 32.06% audited net token saving, compared with 93.0% full-generation accuracy. The observed accuracy loss differs from the nominal budget because policy selection and evaluation use separate data.

E.4 Computation Accounting

Audited saving includes the cost of the original reasoning process and every forced-answer probe actually issued by the policy. For a policy π, the corpus-level saving is defined as

Table 12: Selected parameters and held-out results for five folds on Qwen-7B/MATH-500 under the nominal 1.0% accuracy-loss budget ($m = 3$). Combined saving is calculated from total token counts rather than the average of individual fold savings.

Fold	(ϵ_c, ϵ_h)	γ	θ	Accuracy loss (pp)	Audited saving (%)
0	$(0.005, 0.01)$	1.00	0.702	1.0	19.25
1	$(0.005, 0.01)$	1.00	0.748	-1.0	31.37
2	$(0.005, 0.05)$	1.00	0.887	0.0	33.52
3	$(0.005, 0.05)$	0.25	1.682	2.0	37.12
4	$(0.005, 0.01)$	1.50	0.153	4.0	38.65
Combined	—	—	—	1.2	32.06

$$\mathrm{Save}(\pi) = 1 - \frac{\sum_i K_i^\pi}{\sum_i K_i^F},$$

where K_i^π is the total cost incurred by the stopping policy on question i and K_i^F is the corresponding full-generation cost without stopping-policy probes.

When the policy stops early, its cost includes the prompt, the reasoning prefix through the selected checkpoint, and every probe issued before the exit. When the policy completes the original reasoning trajectory, its cost includes full generation and all probes issued before natural termination. Skipped checkpoints incur no probe cost. The primary Qwen-7B/MATH-500 evaluation uses a full-generation denominator of 2,371,351 tokens.

All reported saving values use the ratio of total token costs across questions. This measure includes the computation spent inspecting checkpoints and therefore reflects net token saving after the cost of monitoring.