STEADYSTOP: RISK-AWARE EARLY STOPPING FOR EFFICIENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long chains of thought can improve problem solving, but models may continue generating after a useful answer is already available. Stopping earlier can reduce this redundancy, yet an interruption may discard reasoning needed to reach a correct answer. Efficient stopping therefore requires distinguishing readiness to answer from the risk of interrupting an otherwise successful trajectory. To address this problem, we propose SteadyStop, a lightweight stopping framework with separately learned *Readiness* and *Hazard* signals. Readiness is trained to predict whether all sampled early answers are correct, while Hazard targets unreliable early answering when full reasoning produces a correct answer. Both signals are learned from reference-checked outputs of the same model using two linear heads with 66 trainable parameters. A *Steadiness* condition reuses Readiness from the preceding anchor, and the first passing anchor triggers one final answer. The reasoning model remains unchanged, with no auxiliary language model required for annotation or online verification. Across two reasoning models and three mathematical benchmarks, SteadyStop reduces output length by 22.86% with a 0.21 percentage point decrease in accuracy, averaged equally over the six settings. This yields a larger mean token reduction and a smaller mean accuracy decrease than the four evaluated baselines.

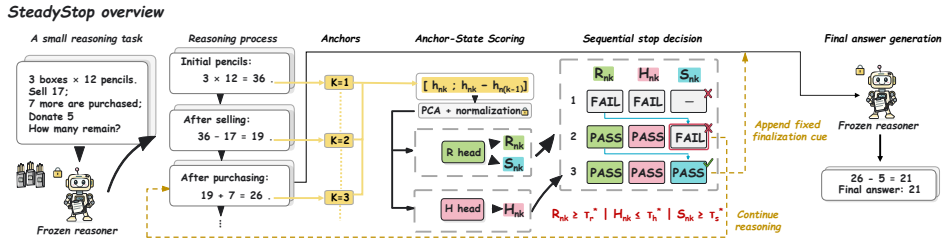

Figure 1: SteadyStop inference. At each anchor, the current hidden state and its change from the preceding anchor feed the Readiness and Hazard heads; preceding Readiness supplies Steadiness. At the first anchor satisfying all three conditions, the model receives a fixed finalization suffix and generates its final answer.

1 INTRODUCTION

Chain-of-Thought reasoning gives language models intermediate computation before they commit to an answer (Wei et al., 2022). Recent reasoning models extend this approach with long trajectories and strong mathematical problem-solving capabilities (Guo et al., 2025; Yang et al., 2025). Yet longer reasoning does not always produce a better answer: a model may repeat an established solution, explore unnecessary alternatives, or revise a correct answer into an incorrect one (Chen et al., 2025; Yang et al., 2026a). Efficient reasoning requires distinguishing such continuation from computation that still helps solve the problem.

Online stopping methods inspect the evolving trajectory through signals such as trial-answer confidence, predicted agreement with the complete answer, or predicted answer convergence (Yang et al., 2026a; Wu et al., 2025; Liu & Wang, 2025). Early stopping also calls for evidence about

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.

the consequences of interruption. An early answer may be unreliable even when the corresponding complete trajectory succeeds, while a reliable early answer can remain useful when the complete answer fails. Agreement between early and complete answers does not establish correctness, since both can be wrong. These cases motivate supervision that uses both early and complete outcomes.

SteadyStop represents these outcomes with separate *Readiness* and *Hazard* targets. Readiness requires all sampled early answers to be correct; Hazard targets unreliable early answering on trajectories with a correct complete answer. The same reasoning model supplies both kinds of answers, checked against references offline. Two linear heads with 66 trainable parameters predict the targets from prefix states. At punctuation and explicit step markers, *Steadiness* reuses the preceding Readiness score to require support at consecutive anchors without a third predictor.

A single offline collection of trajectories, states, and answer outcomes supports head fitting and threshold selection through cached replay. During inference, the controller scores the prefix before asking for an answer; the first passing anchor triggers one final answer and ends generation (Figure 1). It requires access to prefix hidden states but leaves the reasoning model's parameters unchanged. No auxiliary language model is needed for labeling or online verification. Our contributions are:

- **Readiness and Hazard supervision.** We define separate targets for reliable early answering and potentially harmful interruption from the same model's reference-checked early and complete answers.
- **A lightweight stopping framework.** SteadyStop combines two linear heads with 66 trainable parameters and a Steadiness condition that reuses preceding Readiness. Shared offline records support fitting and threshold selection without further answer generation, while the reasoning model remains unchanged.
- **Evaluation across models and reasoning tasks.** Across six mathematical settings, SteadyStop achieves a 22.86% mean token reduction with a 0.21 percentage point decrease in accuracy. Comparisons, ablations, and threshold analysis characterize the stopping rule; transfer experiments reuse each controller without retraining or selecting new thresholds.

2 Related Work

2.1 Test-Time Computation and Concise Reasoning

Test-time computation can support answer aggregation through self-consistency (Wang et al., 2023), search over intermediate thoughts (Yao et al., 2023), and step-level verification (Lightman et al., 2024). Its effective allocation depends on problem difficulty (Snell et al., 2025). Budget-control methods regulate reasoning more directly: TALE places problem-dependent token budgets in the prompt (Han et al., 2025), s1 ends or extends reasoning through budget forcing (Muennighoff et al., 2025), and Dynasor allocates effort using the Certaindex certainty signal (Fu et al., 2025).

Training can instead change the model's generation behavior. C3oT combines original and compressed chains through conditioned training (Kang et al., 2025), and self-training elicits concise reasoning from the model's own sampled responses (Munkhbat et al., 2025). TokenSkip learns controllable compression (Xia et al., 2025), while Self-Braking Tuning teaches the model to stop redundant continuation (Zhao et al., 2025). SteadyStop learns a stopping controller without updating the parameters of the reasoning model, evaluating the evolving prefix rather than prescribing a target response length.

2.2 Early Stopping and Outcome Supervision

DEER uses confidence in trial answers (Yang et al., 2026a). Thought Calibration uses learned probes; its Consistent variant predicts agreement between intermediate and complete answers (Wu et al., 2025). Learn-to-Stop predicts answer convergence with an LSTM (Liu & Wang, 2025). TERMINATOR uses a Transformer probe to predict when the model first derives the answer it would ultimately return (Nagle et al., 2026). Additional signals include attention states in ASAG (Li et al., 2026) and exit prediction combined with speculative decoding in SpecExit (Yang et al., 2026b).

Answer outcomes also supervise stopping. FlashThink's FT^2 verifier learns from early and original answers (Jiang et al., 2025); Conformal Thinking calibrates losses that include missing a future

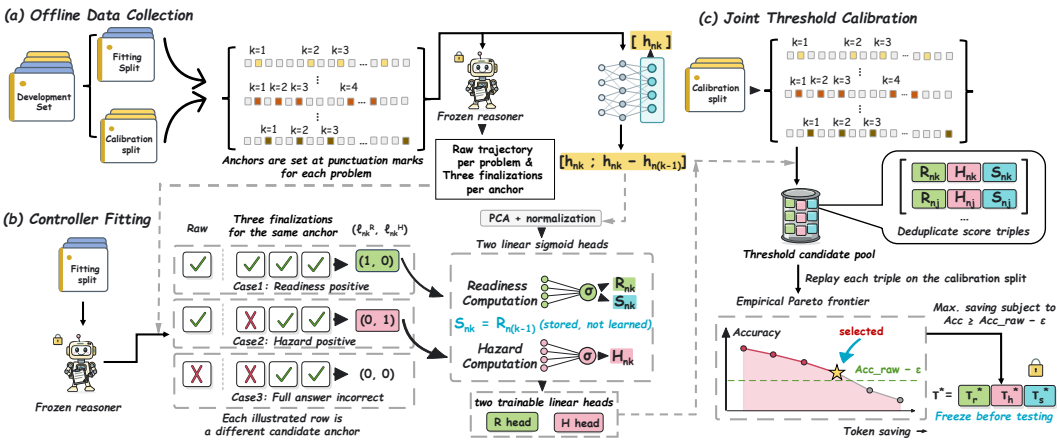

Figure 2: Offline construction and calibration of SteadyStop. (a) A frozen reasoner collects anchored trajectories and three early answers per anchor from a 1,000-problem development set, split into 600 fitting and 400 calibration problems. (b) The fitting split defines Readiness and Hazard labels and trains two linear heads; Steadiness reuses preceding Readiness. (c) Cached replay of deduplicated score triplets maximizes token savings under the accuracy constraint, breaking ties by correct-output count. Online inference is shown in Figure 1.

correct answer (Wang et al., 2026). SteadyStop learns separate Readiness and Hazard targets from reference-checked answers of the same model, then combines both scores with Readiness from the preceding anchor. The original implementations of Thought Calibration and TERMINATOR use auxiliary language models for offline labeling (Wu et al., 2025; Nagle et al., 2026), whereas SteadyStop obtains labels from its reasoning model's own answers. Online, its two heads score prefix features before requesting the final answer, without generating trial answers at each anchor.

3 Methodology

3.1 Overview and Problem Formulation

SteadyStop learns its stopping rule from a shared offline collection of prefix states and answer outcomes (Figure 2). Readiness Computation and Hazard Computation score a prefix, while Steadiness reuses preceding Readiness. Together they determine the first anchor at which the model generates and returns its final answer.

For a problem x_n, *full reasoning* means running the reasoning model under the configured task budget without early stopping. Its output is $y_n^{\text{raw}} = (y_{n1}, \ldots, y_{nT_n})$, with output length T_n. A *prefix* is the part of this output generated so far. Early stopping decides whether to ask for a final answer from that prefix or to continue reasoning.

Anchors in the generated text. For problem x_n, let A_{nk} denote its k-th *anchor*, a checkpoint at token position t_{nk}. Anchors occur when the model completes punctuation or an explicit step marker, such as a comma, period, semicolon, or `Step one`. The matching rule uses only generated text, merges multiple matches at the same position, and does not trigger again merely because an earlier marker remains in the prefix. Normal termination takes precedence over anchor detection, as in Algorithm 1; collection and inference use the same ordering. Anchor positions satisfy $t_{n1} < \cdots < t_{nK_n}$, and their number K_n depends on the generated reasoning. The first anchor initializes score history; later anchors can trigger early stopping.

3.2 Offline Data Collection

We use 1,000 problems, partitioned by problem into disjoint sets $\mathcal{I}_{\text{fit}}$ and $\mathcal{I}_{\text{cal}}$ with 600 fitting and 400 calibration problems. All anchors from a problem remain in the same split. For each problem, we collect a complete reasoning trajectory, its answer correctness c_n^{raw} and output length T_n, and

each anchor's position and raw hidden state before appending the finalization suffix. Early answer sampling uses saved prefixes or key–value (KV) caches and leaves the original trajectory intact.

Every anchor in both splits receives $n_{\text{fin}} = 3$ early-answer samples using the finalization procedure in Section 3.5. We cache each sample's correctness $c_{nk}^{(j)}$ and total output length $C_{nk}^{(j)}$, for $j = 1, 2, 3$. The 600 fitting problems determine the feature transformation and provide labels for the two heads. On the 400 calibration problems, all three outcomes contribute to policy accuracy and mean output length, reducing reliance on a single stochastic answer. After fitting, the frozen transformation and heads score the calibration prefixes once; threshold candidates reuse the same three outcomes without further generation. Three answers at an anchor share one prefix state and do not create three distinct stopping positions.

3.3 Readiness, Hazard, and Steadiness

The two computation modules learn separate binary targets from fitting records and combine their scores in the stopping rule.

Supervision targets. For each fitting anchor, the collection protocol provides n_{fin} sampled early answers; our experiments use $n_{\text{fin}} = 3$. Let $c_{nk}^{(j)} \in \{0, 1\}$ indicate whether sample j is parseable and correct against the reference. The two binary labels are

$$\ell_{nk}^{R} = \mathbf{1}\left[\sum_{j=1}^{n_{\text{fin}}} c_{nk}^{(j)} = n_{\text{fin}}\right], \qquad \ell_{nk}^{H} = c_{n}^{\text{raw}}\big(1 - \ell_{nk}^{R}\big). \tag{1}$$

The Readiness label is positive when every sampled early answer is parseable and correct. The Hazard label is positive when at least one sample fails but the complete answer is correct. The complete outcome distinguishes potentially harmful interruption from unreliable early answering when full reasoning also fails. The labels need not be complementary, but $\ell_{nk}^{R} + \ell_{nk}^{H} \leq 1$.

Reference answers are used offline for supervision and policy evaluation, not as inputs to the computation modules. At inference, both modules read only prefix features. Appendix B.1 analyzes repeated sampling, and Appendix B.3 distinguishes the labels' conditional probabilities from the learned scores.

Prefix features. Let $\mathbf{h}_{nk} \in \mathbb{R}^{d_h}$ be the last-layer state at the final token of the reasoning prefix at A_{nk}. We combine it with the change from the preceding anchor:

$$\Delta\mathbf{h}_{nk} = \begin{cases} \mathbf{0}, & k = 1, \\ \mathbf{h}_{nk} - \mathbf{h}_{n(k-1)}, & k \geq 2, \end{cases} \quad \mathbf{u}_{nk} = [\mathbf{h}_{nk}; \Delta\mathbf{h}_{nk}], \quad \mathbf{z}_{nk} = F(\mathbf{u}_{nk}). \tag{2}$$

Throughout, n indexes problems and k indexes anchors; $n(k-1)$ refers to the preceding anchor of problem n. Vectors are columns, $[\cdot;\cdot]$ denotes concatenation, and $\mathbf{u}_{nk} \in \mathbb{R}^{2d_h}$ and $\mathbf{z}_{nk} \in \mathbb{R}^{d}$. The transform F consists of principal component analysis (PCA) and normalization fitted only on the fitting portion and then fixed, with $d = 32$ in our experiments. Appendix B.2 specifies the transformation.

Readiness Computation and Hazard Computation. The two modules f_R and f_H output

$$\begin{aligned} R_{nk} &= f_R(\mathbf{z}_{nk}) = \sigma(\mathbf{w}_R^{\top}\mathbf{z}_{nk} + b_R), \\ H_{nk} &= f_H(\mathbf{z}_{nk}) = \sigma(\mathbf{w}_H^{\top}\mathbf{z}_{nk} + b_H). \end{aligned} \tag{3}$$

The weights $\mathbf{w}_R, \mathbf{w}_H \in \mathbb{R}^d$ and scalar biases are trainable; σ denotes the sigmoid function. These outputs are learned scores, with thresholds selected by the resulting policy's calibration performance. Define $\mathcal{I}_{\text{fit}}^{+} = \{n \in \mathcal{I}_{\text{fit}} : K_n > 0\}$ and $N_{\text{fit}}^{+} = |\mathcal{I}_{\text{fit}}^{+}|$. For this nonempty set, we train with equal total weight per problem:

$$\mathcal{L} = \frac{1}{N_{\text{fit}}^{+}} \sum_{n \in \mathcal{I}_{\text{fit}}^{+}} \frac{1}{K_n} \sum_{k=1}^{K_n} \left[\text{BCE}(R_{nk}, \ell_{nk}^{R}) + \text{BCE}(H_{nk}, \ell_{nk}^{H})\right]. \tag{4}$$

Here BCE denotes binary cross entropy. Averaging within each problem prevents trajectories with more anchors from dominating training. Because the heads have distinct parameters and the feature transform is fixed, each loss updates only its own head. Their total trainable parameter count is $2(d+1)$; optimization details are in Appendix A.

Steadiness and stopping conditions. Steadiness reuses Readiness from the immediately preceding anchor: $S_{nk} = R_{n(k-1)}$ for $k \geq 2$. Together with current Readiness, it requires support for answering at consecutive anchors under the respective thresholds τ_R and τ_S. It introduces no additional head or training target. With high Readiness, low Hazard, and high Steadiness, the stopping rule is

$$g_{nk}(\boldsymbol{\tau}) = \begin{cases} \mathbf{1}[R_{nk} \geq \tau_R \ \wedge \ H_{nk} \leq \tau_H \ \wedge \ S_{nk} \geq \tau_S], & k \geq 2, \\ 0, & k = 1. \end{cases} \tag{5}$$

Here $\boldsymbol{\tau} = (\tau_R, \tau_H, \tau_S)$. The first anchor initializes history but cannot trigger stopping. Every later check reads preceding Readiness before updating history, including after a failed check. Its threshold τ_S may differ from τ_R, and the preceding anchor need not have passed its own gate. Steadiness adds no learned parameters. Fixed preprocessing can also be folded into the two heads, yielding two affine scores and three comparisons per check (Appendix B.2); the trainable parameter count excludes fixed preprocessing storage.

3.4 Threshold Selection

After fitting, the fixed feature transformation and computation modules score every stored anchor of the $N_{\text{cal}} = 400$ calibration problems. We obtain R_{nk}, H_{nk}, and, for $k \geq 2$, S_{nk}. Each anchor retains three cached correctness and output-length pairs $(c_{nk}^{(j)}, C_{nk}^{(j)})$. Adding the scores requires feature and head computation, but no additional answer generation.

Candidate thresholds and replay. Each complete observed score triple provides a candidate threshold vector:

$$\Theta_{\text{cal}} = \{(R_{nk}, H_{nk}, S_{nk}) : n \in \mathcal{I}_{\text{cal}},\ 2 \leq k \leq K_n\}. \tag{6}$$

The set removes duplicate triples while keeping their coordinates together. Every candidate is applied globally to all calibration problems, rather than forming a grid from separate score values. For each $\boldsymbol{\tau}$, replay selects the first anchor that satisfies the stopping conditions:

$$k_n(\boldsymbol{\tau}) = \min(\{k : 2 \leq k \leq K_n,\ g_{nk}(\boldsymbol{\tau}) = 1\} \cup \{\infty\}). \tag{7}$$

When this index is finite, we retrieve all three outcomes at that anchor; otherwise, each sample slot uses the full-reasoning result. Denote the replayed values by $c_n^{(j)}(\boldsymbol{\tau})$ and $C_n^{(j)}(\boldsymbol{\tau})$. Scores select one common exit per problem before correctness is read. Thus an anchor with two correct answers contributes two successes out of three, not a majority-vote success or a failed unanimity label. The label ℓ_{nk}^R is used for fitting, not for calibration accuracy.

Minimizing output length under an accuracy constraint. Policy metrics average over the $3N_{\text{cal}}$ replayed outcomes, including normal completions:

$$\text{Acc}_{\text{cal}}(\boldsymbol{\tau}) = \frac{\sum_{n \in \mathcal{I}_{\text{cal}}} \sum_{j=1}^{3} c_n^{(j)}(\boldsymbol{\tau})}{3N_{\text{cal}}}, \qquad \text{Tok}_{\text{cal}}(\boldsymbol{\tau}) = \frac{\sum_{n \in \mathcal{I}_{\text{cal}}} \sum_{j=1}^{3} C_n^{(j)}(\boldsymbol{\tau})}{3N_{\text{cal}}}.$$

The full-reasoning reference is $\text{Acc}_{\text{cal}}^{\text{raw}} = N_{\text{cal}}^{-1} \sum_{n \in \mathcal{I}_{\text{cal}}} c_n^{\text{raw}}$. For a preset tolerance $\varepsilon \geq 0$, we retain candidates within ε of this accuracy or better, minimize tokens, and break ties by maximizing the total number of correct outputs:

$$\begin{aligned} \Theta_\varepsilon &= \{\boldsymbol{\tau} \in \Theta_{\text{cal}} : \text{Acc}_{\text{cal}}(\boldsymbol{\tau}) \geq \text{Acc}_{\text{cal}}^{\text{raw}} - \varepsilon\}, \\ \Theta_\varepsilon^{\text{tok}} &= \arg\min_{\boldsymbol{\tau} \in \Theta_\varepsilon} \text{Tok}_{\text{cal}}(\boldsymbol{\tau}), \\ \boldsymbol{\tau}^\star &\in \arg\max_{\boldsymbol{\tau} \in \Theta_\varepsilon^{\text{tok}}} \sum_{n \in \mathcal{I}_{\text{cal}}} \sum_{j=1}^{3} c_n^{(j)}(\boldsymbol{\tau}). \end{aligned} \tag{8}$$

Accuracy and ε use the $[0, 1]$ scale; reported percentages multiply accuracy by 100. Minimizing output length is equivalent to maximizing token reduction against the fixed full-reasoning reference. The selected thresholds are fixed for all evaluation problems and tasks. If Θ_ε is empty, early stopping is disabled and the model uses full reasoning. Appendix A.3 specifies replay and tie-breaking.

Algorithm 1 Online inference with SteadyStop

Require: Problem x_n, fixed reasoning model, transform F, Readiness Computation f_R, Hazard Computation f_H, ISBOUNDARY, thresholds $\boldsymbol{\tau}^\star$, generation settings

$k \leftarrow 0$; $(\mathbf{h}_{\text{prev}}, R_{\text{prev}}) \leftarrow (\varnothing, \varnothing)$
while ordinary reasoning generation has not ended **do**
  Generate the next token and update the normal decoding state
  if normal generation has ended **then**
    return the normally completed answer
  end if
  if ISBOUNDARY(current prefix) is false **then**
    continue
  end if
  $k \leftarrow k + 1$; define checkpoint A_{nk} at the current token position t_{nk}
  Read $\mathbf{h}_{nk}$ from the current reasoning prefix
  $\Delta\mathbf{h}_{nk} \leftarrow \mathbf{0}$ if $k = 1$, otherwise $\mathbf{h}_{nk} - \mathbf{h}_{\text{prev}}$
  $\mathbf{z}_{nk} \leftarrow F([\mathbf{h}_{nk}; \Delta\mathbf{h}_{nk}])$
  $R_{nk} \leftarrow f_R(\mathbf{z}_{nk})$; $H_{nk} \leftarrow f_H(\mathbf{z}_{nk})$
  stop $\leftarrow$ false
  if $k \geq 2$ **then**
    $S_{nk} \leftarrow R_{\text{prev}}$
    stop $\leftarrow (R_{nk} \geq \tau_R^\star) \wedge (H_{nk} \leq \tau_H^\star) \wedge (S_{nk} \geq \tau_S^\star)$
  end if
  $(\mathbf{h}_{\text{prev}}, R_{\text{prev}}) \leftarrow (\mathbf{h}_{nk}, R_{nk})$
  if stop **then**
    Append the finalization suffix and generate one final answer with the same model
    return the generated final answer
  end if
end while
return the normally completed answer

3.5 INFERENCE

At the first anchor satisfying the three conditions under $\boldsymbol{\tau}^\star$, we append a finalization suffix, and the model generates one final answer. Otherwise, generation continues normally. The generated answer is returned directly, without an acceptance check, retry, or resumption of reasoning (Algorithm 1).

Final answer generation. The finalization suffix depends on whether the prefix's thinking region is still open. It closes an open region when needed; otherwise, no closing tag is included:

Still inside `<think>`

```
</think>
Final Answer:  \boxed{
```

No open thinking region

```
Final Answer:  \boxed{
```

The second case covers prefixes after `</think>` or without a thinking span. The model completes the open brace with its answer; the suffix supplies no answer content. Both offline splits use this rule for three sampled completions, and online exits use it for one. Generation limits appear in Section 4.1.

4 EXPERIMENTS

We evaluate accuracy and token count on two reasoning models, then examine the stopping conditions, threshold sensitivity, and transfer without controller retraining.

4.1 EXPERIMENTAL SETUP

Models and benchmarks. We use Qwen3-1.7B (Yang et al., 2025) and DeepSeek-R1-Distill-Qwen-14B (Guo et al., 2025). Mathematical benchmarks are GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024; Hugging Face H4, 2024), and AIMO Validation AMC (AI-MO, 2024). Each run evaluates their complete sets of 1,319, 500, and 83 problems, respectively.

Method	Venue	GSM8K			MATH-500			AMC (10,240)		
		Acc.↑	Tokens↓	Red.↑	Acc.↑	Tokens↓	Red.↑	Acc.↑	Tokens↓	Red.↑
Qwen3-1.7B										
Full reasoning	–	80.74	1,821.93	0.00	82.00	4,363.73	0.00	66.27	8,205.89	0.00
TokenSkip	EMNLP (2025)	77.26	1,255.63	31.08	79.13	3,274.86	24.95	62.25	7,346.21	10.48
SBT-D	NeurIPS (2025)	81.17	1,401.28	23.09	78.87	3,204.57	26.56	62.65	6,727.94	18.01
DEER	ICLR (2026a)	81.12	1,293.47	29.01	80.40	3,154.79	27.70	64.26	6,808.13	17.03
T-Calibration	EMNLP (2025)	80.06	1,568.92	13.89	80.60	3,638.41	16.62	64.66	7,014.68	14.52
SteadyStop	Ours	84.84	1,137.08	37.59	80.80	3,110.32	28.72	65.06	6,671.48	18.70
DeepSeek-R1-Distill-Qwen-14B										
Full reasoning	–	83.90	804.82	0.00	89.80	3,033.00	0.00	80.72	5,438.10	0.00
TokenSkip	EMNLP (2025)	84.05	719.46	10.61	86.40	2,624.91	13.45	77.11	4,086.72	24.85
SBT-D	NeurIPS (2025)	83.24	696.83	13.42	85.33	2,677.36	11.73	76.31	4,469.27	17.82
DEER	ICLR (2026a)	83.47	702.84	12.67	87.47	2,640.73	12.93	77.91	4,389.61	19.28
T-Calibration	EMNLP (2025)	83.67	748.27	7.03	86.93	2,789.18	8.04	77.51	4,562.38	16.10
SteadyStop	Ours	85.97	677.57	15.81	87.20	2,557.08	15.69	78.31	4,315.19	20.65

Table 1: Main mathematical results (three-run means). Acc. and Red. are percentages, with Red. measured relative to full reasoning. The value in parentheses after AMC is the generation-token budget. T-Calibration denotes the Consistent variant; all methods use the settings in Section 4.1.

AMC reformulates 2022–2023 AMC 12 questions to have integer answers. Appendix A lists the remaining configuration details.

SteadyStop fitting and calibration. For each model, we use Competition MATH problems for the collection and fitting procedure in Section 3.2, with a 12,288 token generation budget and $d = 32$. Each model has its own controller and thresholds, fixed before evaluation. Appendix A gives the sampling and optimization settings.

Generation settings. Both models use temperature 0.60, top-p 0.95, and top-k 20. Token budgets are 4,096 for GSM8K, CommonsenseQA, and ARC-Challenge, 8,192 for MATH-500, and 10,240 for AMC; the same budgets apply to ablations and threshold analysis. Early-answer generation is capped at 16 new tokens per answer during offline collection and online stopping. For both models, threshold selection uses $\varepsilon = 0.02$, allowing at most a 2 percentage point accuracy decrease on the calibration split relative to full reasoning. The calibrated $(\tau_R^\star, \tau_H^\star, \tau_S^\star)$ is $(0.83, 0.19, 0.77)$ for Qwen3-1.7B and $(0.74, 0.24, 0.82)$ for DeepSeek-R1-Distill-Qwen-14B. Thresholds are displayed to two decimals; selection and score comparisons retain full precision.

Baselines. We compare with TokenSkip (Xia et al., 2025) and SBT-D (Zhao et al., 2025), which adapt the reasoning model, and DEER (Yang et al., 2026a) and the Consistent variant of Thought Calibration (T-Calibration; Wu et al., 2025), which control stopping during inference. All methods start from the corresponding original checkpoint and use the same evaluation problems, tokenizer, task budgets, common decoding settings, and three-run aggregation. We retain each method's required prompts, training procedure, and stopping mechanism. TokenSkip uses a compression ratio of 0.70, and SBT-D uses an overthinking threshold of 0.20. External baselines appear only in the main mathematical comparison; Appendix A.4 describes their method-specific settings.

Metrics. We report mean accuracy (Acc., %) and generated output length (Tokens) over three runs. Generated output length counts the reasoning prefix and final answer; normal completion uses the full generated output length. The appended finalization suffix is input context, not generated output. The same accounting convention applies to every method. Unreturned trial answers are excluded, so this metric is not end-to-end execution cost. Unparseable answers count as incorrect; no failed answers are discarded. Red. is the percentage decrease in mean tokens relative to full reasoning under the same task budget, and accuracy changes are in percentage points (pp). Aggregate comparisons use an

Configuration	GSM8K		MATH-500		AMC	
	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓
Full reasoning	80.74	1,821.93	82.00	4,363.73	66.27	8,205.89
SteadyStop	84.84	1,137.08	80.80	3,110.32	65.06	6,671.48
Without Readiness	84.69	1,085.76	78.20	3,049.20	61.45	5,339.83
Without Hazard	83.93	1,151.36	79.00	3,207.33	66.27	6,087.42
Without Steadiness	80.14	786.58	74.00	2,173.43	53.01	4,211.35

Table 2: Stopping-condition ablations on Qwen3-1.7B (three-run means). Each variant removes one gate condition while retaining the fitted heads and prefix features; task budgets and the full-rule reference match Table 1.

unweighted mean over the six model–benchmark settings. Differences and ratios use stored source values, with final displays rounded to two decimal places.

4.2 Main Results

SteadyStop shortens outputs in all six mathematical settings, averaging a 22.86% reduction with a 0.21 pp accuracy decrease. On GSM8K, Qwen3-1.7B gains 4.09 pp with 37.59% fewer tokens, and the 14B model gains 2.07 pp with a 15.81% reduction. On MATH-500 and AMC, both controllers trade lower accuracy for shorter outputs (Table 1).

Comparison with the baselines. SteadyStop achieves both higher accuracy and fewer output tokens in 22 of the 24 comparisons against the four baselines (four methods across six settings). The two exceptions illustrate accuracy–length trade-offs: on 14B AMC, TokenSkip produces shorter outputs, while SteadyStop is 1.20 pp more accurate; on 14B MATH-500, DEER is 0.27 pp more accurate but uses 83.65 more tokens per problem. Overall, SteadyStop provides the largest mean token reduction and the smallest mean accuracy decrease among the five efficiency methods.

4.3 Ablation Study

We remove Readiness, Hazard, or Steadiness from the stopping rule while retaining the prefix features and the two fitted computation modules. The resulting rules use (H, S), (R, S), and (R, H), respectively, under the budgets in Section 4.1. Removing Steadiness retains $\Delta\mathbf{h}$.

Readiness. Removing the current Readiness check saves 51.32, 61.12, and 1,331.65 tokens on GSM8K, MATH-500, and AMC, respectively, but lowers accuracy by 0.15, 2.60, and 3.61 pp. Readiness evaluates the current prefix, whereas Steadiness refers to the preceding anchor. The results favor checking both positions rather than relying on preceding Readiness and low Hazard alone.

Hazard. On GSM8K and MATH-500, the complete rule is 0.91 and 1.80 pp more accurate than the rule without Hazard and uses 14.28 and 97.01 fewer tokens, respectively. Thus, the Hazard score improves both reported metrics beyond the two Readiness checks on these tasks. Unlike Readiness, its supervision uses the full-reasoning outcome to distinguish unreliable early answers associated with a correct complete answer. On AMC, the rule without Hazard is 1.20 pp more accurate and uses 584.06 fewer tokens, showing that the benefit depends on the task.

Steadiness. Removing Steadiness retains $\Delta\mathbf{h}$ but no longer requires preceding Readiness to pass its threshold. This variant has the shortest outputs among the three ablations, but also the largest accuracy decreases: 4.70, 6.80, and 12.05 pp on GSM8K, MATH-500, and AMC, respectively. The complete rule still reduces tokens by 37.59%, 28.72%, and 18.70%, providing a different quality–length trade-off through explicit confirmation across consecutive anchors.

A concrete stopping example. Appendix C traces an AMC problem whose shared prefix has already derived $109/33$ and the answer 142. At a later punctuation boundary, SteadyStop returns the same correct answer using 1,301 rather than 2,738 generated tokens (52.48% fewer); the uninterrupted continuation repeats arithmetic, divisor checks, and the answer presentation. The example illustrates what the complete controller removes after an answer has been established.

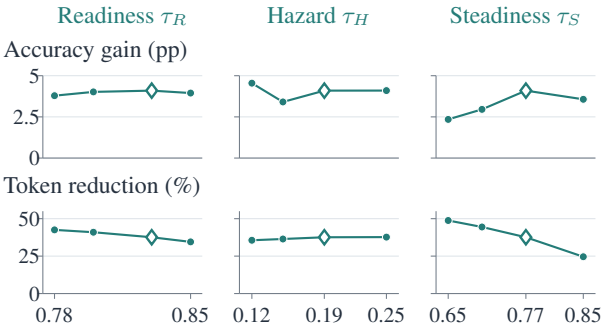

Figure 3: Threshold sweeps on GSM8K (Qwen3-1.7B). Changes are relative to full reasoning. Diamonds mark calibrated values; other thresholds stay fixed.

Task	Accuracy (%) Full	SteadyStop	Red. (%)
Qwen3-1.7B			
CommonsenseQA	73.46	70.93	**43.82**
ARC-Challenge	88.48	85.67	**46.85**
DeepSeek-R1-Distill-Qwen-14B			
CommonsenseQA	81.33	81.08	**14.05**
ARC-Challenge	94.88	93.94	**14.44**

Table 3: Transfer with fixed controllers. Accuracy and token reduction are percentages.

4.4 Threshold Analysis

Figure 3 varies one threshold at a time on GSM8K while holding the other two fixed. Accuracy ranges from 83.09% to 85.29%, above the 80.74% full-reasoning reference, while token reductions range from 24.58% to 48.79%. The joint quality–length improvement therefore persists beyond the calibrated setting. Higher Readiness and Steadiness thresholds accompany smaller token reductions, whereas relaxing the Hazard threshold produces a narrower range of savings. Accuracy varies non-monotonically, so stricter checks do not necessarily yield better aggregate results. On MATH-500, increasing τ_S from 0.65 to 0.85 raises accuracy from 78.20% to 83.00%, while token reduction decreases from 37.36% to 15.77%. The thresholds thus specify exit evidence rather than output length. Main results use thresholds selected jointly on the calibration split, not retuned on evaluation tasks.

4.5 Generalization Beyond Mathematics

Without retraining or threshold changes, we evaluate each frozen controller on all 1,221 examples in the CommonsenseQA validation split (Talmor et al., 2019) and all 1,172 examples in the ARC-Challenge test split (Clark et al., 2018). Qwen3-1.7B reduces tokens by 43.82% and 46.85%, with accuracy decreases of 2.54 and 2.82 pp; the 14B model reduces tokens by 14.05% and 14.44%, with decreases of 0.25 and 0.94 pp (Table 3). The two models retain different accuracy–length trade-offs under transfer: Qwen3-1.7B nearly halves its outputs, while the 14B controller preserves accuracy to within one percentage point. Both controllers were fitted and calibrated only on mathematics, then reused without task-specific retraining or threshold selection. Detailed results, including mean token counts, appear in Appendix D.

5 Limitations

SteadyStop requires access to hidden states, limiting its use with black-box APIs. Computational constraints restricted our experiments to models with at most 14B parameters. We expect the approach to extend to larger models because its fitting and stopping procedure is not specific to a model size, but this remains to be verified empirically.

6 Conclusion

SteadyStop combines learned Readiness and Hazard with preceding Readiness to select the first anchor satisfying its stopping conditions. The controller has 66 trainable parameters; fitting and calibration leave the reasoning model unchanged and require no auxiliary language model. Across six mathematical settings, it reduces generated output length by 22.86% on average with a 0.21 pp decrease in macro-averaged accuracy. The same controllers also transfer to other reasoning tasks without retraining or threshold changes.

AI USE STATEMENT

In preparing this paper, we used generative AI tools to aid and polish the writing, including improving grammar, clarity, and readability. We have reviewed and edited the AI-assisted text and take responsibility for the final content of this work, including its text, claims, and artifacts.

REFERENCES

AI-MO. AIMO Validation AMC. Hugging Face dataset repository, 2024. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc. 83 integer-answer adaptations of AMC12 problems from 2022 and 2023; accessed September 17, 2026.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? On the overthinking of long reasoning models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 9487–9499. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/chen25bx.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. 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.

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 with Certaindex. In *Advances in Neural Information Processing Systems*, volume 38, pp. 141987–142025, 2025. doi: 10.52202/085713-4751. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d037fd021c9aace128b8ce25001cdb6c-Abstract-Conference.html.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025. doi: 10.18653/v1/2025.findings-acl.1274. URL https://aclanthology.org/2025.findings-acl.1274/.

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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Hugging Face H4. MATH-500. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/HuggingFaceH4/MATH-500. A 500-problem MATH test subset released by OpenAI with Let's Verify Step by Step; accessed September 17, 2026.

Guochao Jiang, Guofeng Quan, Zepeng Ding, Ziqin Luo, Dixuan Wang, and Zheng Hu. FlashThink: An early exit method for efficient reasoning. *arXiv preprint arXiv:2505.13949*, 2025. URL https://arxiv.org/abs/2505.13949.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3oT: Generating shorter chain-of-thought without compromising effectiveness. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24312–24320, 2025. doi: 10.1609/aaai.v39i23.34608. URL https://ojs.aaai.org/index.php/AAAI/article/view/34608.

Jiakai Li, Ke Qin, Rongzheng Wang, Yizhuo Ma, Qizhi Chen, Muquan Li, and Shuang Liang. Stop when further reasoning won't help: Attention-state adaptive generation in reasoning models. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/60517.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

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, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.904. URL https://aclanthology.org/2025.emnlp-main.904/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Fei-Fei Li, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. Self-training elicits concise reasoning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25127–25152, 2025. doi: 10.18653/v1/2025.findings-acl.1289. URL https://aclanthology.org/2025.findings-acl.1289/.

Alliot Nagle, Jakhongir Saydaliev, Dhia Garbaya, Michael Gastpar, Ashok Vardhan Makkuva, and Hyeji Kim. TERMINATOR: Learning optimal exit points for early stopping in chain-of-thought reasoning. *arXiv preprint arXiv:2603.12529*, 2026. doi: 10.48550/arXiv.2603.12529. URL https://arxiv.org/abs/2603.12529.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Xi Wang, Anushri Suresh, Alvin Zhang, Rishi More, William Jurayj, Benjamin Van Durme, Mehrdad Farajtabar, Daniel Khashabi, and Eric Nalisnick. Conformal thinking: Risk control for reasoning on a compute budget. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/61680.

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 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. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Menghua Wu, Cai Zhou, Stephen Bates, and Tommi Jaakkola. Thought Calibration: Efficient and confident test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14291–14305, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.722. URL https://aclanthology.org/2025.emnlp-main.722/.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.165. URL https://aclanthology.org/2025.emnlp-main.165/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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*, pp. 88170–88210, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8df90a1440ce782d1f5607b7a38f2531-Abstract-Conference.html.

Rubing Yang, Huajun Bai, Song Liu, Guanghua Yu, Runzhi Fan, Yanbin Dang, Jiejing Zhang, Kai Liu, Jianchen Zhu, and Peng Chen. SpecExit: Accelerating large reasoning model via speculative exit. In *International Conference on Machine Learning*, 2026b. URL https://icml.cc/virtual/2026/poster/66249.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Haoran Zhao, Yuchen Yan, Yongliang Shen, Haolei Xu, Wenqi Zhang, Kaitao Song, Jian Shao, Weiming Lu, Jun Xiao, and Yueting Zhuang. Let LRMs break free from overthinking via self-braking tuning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 1861–1887, 2025. doi: 10.52202/085713-0069. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/02d028c3bf5d4875b5a4e4b4311087f7-Abstract-Conference.html.

A Implementation and Reproducibility Details

A.1 SteadyStop Data and Training Configuration

Data partitions. Each model has a shared offline collection of 1,000 Competition MATH problems, partitioned by problem into 600 fitting and 400 calibration records. Only the fitting portion determines PCA, normalization, and the two heads. Dataset sources are `hendrycks/competition_math`, `openai/gsm8k`, `HuggingFaceH4/MATH-500`, `AI-MO/aimo-validation-amc`, `tau/commonsense_qa`, and `allenai/ai2_arc` on Hugging Face.

Offline collection records. The cache retains complete trajectories and answers, correctness c_n^{raw}, and output length T_n. In both splits, each A_{nk} links its position t_{nk} and raw prefix state $\mathbf{h}_{nk}$ to three early-answer samples, together with their correctness and output lengths. States are recorded before the finalization suffix is appended. Calibration scores are added after the transformation and heads have been fitted.

Generation and evaluation. The reasoning models remain fixed during collection, fitting, calibration, and evaluation. Table 4 summarizes the settings. Three independent evaluation runs determine the reported means. Each offline anchor has three early answers: fitting combines the all-correct condition with the full reasoning outcome to construct both labels, while calibration averages the three cached outcomes. An online exit generates only one final answer. All early answers are limited to 16 new tokens. Output lengths follow Section 4.1. Transfer evaluation uses the complete CommonsenseQA validation split (1,221 examples) and ARC-Challenge test split (1,172 examples).

Features and optimization. Both heads use the last-layer state of the last prefix token and its change from the preceding anchor. We implement Equation 4 as numerically stable weighted BCE-with-logits on the affine scores, with equal total weight per problem, and optimize with AdamW (Loshchilov & Hutter, 2019). The reasoning model and fitted transformation remain fixed during optimization. First anchors participate with $\Delta\mathbf{h}_{n1} = \mathbf{0}$; problems with no anchors complete normally during calibration and evaluation. The controller stores the preceding hidden state and Readiness score, and its fixed transformation admits $O(d_h)$ scoring once states are available (Appendix B.2).

Parameter	Value
Common temperature / top-p / top-k	0.60 / 0.95 / 20
Independent evaluation runs	3
Offline collection token budget	12,288
Collection / fitting / calibration problems	1,000 / 600 / 400
Early answers per fitting anchor (n_{fin})	3
Early answers per calibration anchor	3
Calibration accuracy-loss tolerance (ε)	0.02 (2 pp)
PCA dimension (d)	32
Trainable parameters in the two heads	66
Objective	Weighted BCE with logits
Optimizer / updates	AdamW / 500
Learning rate / weight decay	0.02 / 0.05
Online final answers per exit	1
Maximum new tokens per early answer	16

Table 4: Decoding and evaluation settings (top) and SteadyStop training and calibration settings (bottom), shared by both models. The 66 trainable parameters belong only to the two linear heads.

A.2 Online Execution

IsBoundary recognizes newly completed punctuation and explicit step markers, merging matches at the same position. Normal termination takes precedence over checking. The first anchor initializes history, and every anchor updates it, even when the gate fails. At later anchors, the preceding

Readiness score is read before the update. Consecutive anchors need not be adjacent tokens or a fixed distance apart.

A.3 Calibration and Ablation Protocol

Scoring stored calibration records. After fitting, the fixed transform and two heads score the 400 calibration problems' stored prefixes in anchor order. The first anchor's Readiness supplies history for the second; $S_{nk} = R_{n(k-1)}$ for $k \geq 2$. Each anchor already has three cached early-answer outcomes. Scoring and replay reuse these records without rerunning the reasoning model. The three samples share the same anchor features and scores, so they do not enlarge the candidate threshold pool or create new anchors.

Cached results. For the index selected by Equation 7, replay retrieves each sample $j \in \{1, 2, 3\}$:

$$(c_n^{(j)}(\boldsymbol{\tau}), C_n^{(j)}(\boldsymbol{\tau})) = \begin{cases} (c_{n(k_n(\boldsymbol{\tau}))}^{(j)}, C_{n(k_n(\boldsymbol{\tau}))}^{(j)}), & k_n(\boldsymbol{\tau}) < \infty, \\ (c_n^{\mathrm{raw}}, T_n), & k_n(\boldsymbol{\tau}) = \infty. \end{cases} \tag{A.1}$$

Each problem contributes three equally weighted outcomes at one selected exit. Three correct samples contribute three successes; two contribute two, and so on. Averaging over $3 \times 400 = 1{,}200$ outcomes is equivalent to averaging the three-sample accuracies of the 400 problems. It is not an average over every cached anchor, which would overweight long trajectories, and is neither pass@3 nor majority voting. For normal completions, the same raw outcome fills all three slots, preserving equal problem weights. Token cost averages the three possible returned outputs rather than charging all three offline samples to one online answer. The fitting label ℓ_{nk}^R remains distinct from this empirical accuracy.

Threshold search. Equation 8 first minimizes mean output length among candidates whose calibration accuracy is within ε of, or above, the full-reasoning reference. We use $\varepsilon = 0.02$ for both models. Among candidates tied for the largest token reduction, we select the one with the most correct outputs across the $3 \times 400 = 1{,}200$ replayed outcomes. This count sums the three correctness indicators at each problem's selected exit, with the same full-reasoning fallback for normal completions; it does not count all-correct anchors or pool all cached anchors. With a fixed denominator, maximizing this count is equivalent to maximizing calibration accuracy, but only among token-optimal feasible candidates. The original score precision is used for constructing the candidate set and for comparisons. Let $M_{\mathrm{cal}} = \sum_{n \in \mathcal{I}_{\mathrm{cal}}} K_n$. A direct scan costs $O(|\Theta_{\mathrm{cal}}| M_{\mathrm{cal}})$ comparisons in the worst case and reuses cached model outputs. If no candidate satisfies the accuracy constraint, early stopping is disabled and the model uses full reasoning.

Ablation scope. The variants remove gate conditions while retaining the original features and both fitted heads, including $\Delta\mathbf{h}$ when Steadiness is removed. They compare stopping rules rather than removing supervision targets.

A.4 External Baseline Configurations

The four external baselines use the common evaluation protocol in Section 4.1. We preserve method-specific data construction, training, prompts, and stopping rules; SteadyStop's projection dimension and head optimization settings do not apply to the baselines.

TokenSkip and SBT-D. TokenSkip (Xia et al., 2025) learns from compressed reasoning with compression ratio 0.70. SBT-D (Zhao et al., 2025) uses the dynamic overthinking criterion with threshold 0.20 for training data construction. Both adapt the corresponding original model through supervised fine tuning, retaining their own masking and prompting procedures.

DEER. DEER (Yang et al., 2026a) retains its own candidate checks, trial answers, and confidence criterion, rather than substituting SteadyStop anchors or scores. Its required answer prompts and checking mechanism are method-specific.

T-Calibration (Consistent). The Consistent variant (Wu et al., 2025) uses a linear probe to predict agreement with the complete answer. We retain its reasoning-step features, auxiliary annotation procedure, and calibration mechanism for the corresponding target model.

B Additional Derivations

B.1 Repeated Sampling and Conservative Labels

Fix a fitting prefix and the correctness of its full reasoning result, suppressing their indices. Let $q \in [0, 1]$ be the probability that an early answer is parseable and correct under a fixed prompt and decoding configuration. Assume the n_{fin} samples are independent draws from this same conditional distribution. The labels in Equation 1 then satisfy

$$\Pr(\ell^R = 1) = q^{n_{\mathrm{fin}}}, \qquad \mathbb{E}[\ell^H] = c^{\mathrm{raw}}(1 - q^{n_{\mathrm{fin}}}). \tag{B.1}$$

The first identity is the probability that all samples succeed. The second follows by taking the expectation of $\ell^H = c^{\mathrm{raw}}(1 - \ell^R)$ with the complete answer fixed. If $c^{\mathrm{raw}} = 1$ and the single-answer failure probability is at least $\delta \in [0, 1]$, then

$$\Pr(\ell^H = 0) = q^{n_{\mathrm{fin}}} \le (1 - \delta)^{n_{\mathrm{fin}}}. \tag{B.2}$$

More samples make a positive Readiness label less likely and provide more opportunities to observe a failure when full reasoning is correct. With $n_{\mathrm{fin}} = 3$, the label probabilities involve q^3 and $c^{\mathrm{raw}}(1 - q^3)$; the failure probability of one answer sampled under the same conditional distribution is instead $1 - q$. These are properties of sampled labels with the complete outcome fixed, not calibration guarantees for the learned scores or the deployed stopping rule.

B.2 Fixed Preprocessing and Equivalent Scoring

PCA and normalization. The transform in Equation 2 is

$$F(\mathbf{u}) = \left(P^\top(\mathbf{u} - \boldsymbol{\mu}_u) - \boldsymbol{\mu}_z\right) \oslash \mathbf{s}_z. \tag{B.3}$$

Here $P \in \mathbb{R}^{2d_h \times d}$ is the PCA projection, $\boldsymbol{\mu}_u \in \mathbb{R}^{2d_h}$ is the input mean, and $\boldsymbol{\mu}_z, \mathbf{s}_z \in \mathbb{R}^d$ are the projected mean and scale. Every component of $\mathbf{s}_z$ is positive, and $\oslash$ denotes elementwise division. These quantities are estimated only on the 600 fitting problems and then fixed.

Folding the fixed transform. Let $D_z = \mathrm{diag}(\mathbf{s}_z)$. For either module $v \in \{R, H\}$, define

$$\widetilde{\mathbf{w}}_v = P D_z^{-1} \mathbf{w}_v, \qquad \widetilde{b}_v = b_v - \mathbf{w}_v^\top D_z^{-1}(P^\top \boldsymbol{\mu}_u + \boldsymbol{\mu}_z). \tag{B.4}$$

Substituting Equation B.3 gives $\mathbf{w}_v^\top F(\mathbf{u}) + b_v = \widetilde{\mathbf{w}}_v^\top \mathbf{u} + \widetilde{b}_v$, with $\widetilde{\mathbf{w}}_v \in \mathbb{R}^{2d_h}$. Applying sigmoid therefore produces the same Readiness or Hazard score as Equation 3. The folded coefficients are computed from trained weights and fixed statistics; they are not additional learned parameters.

Comparing logits. For thresholds in $(0, 1)$, define $\mathrm{logit}(x) = \log[x/(1 - x)]$. Since sigmoid is monotone, the three conditions at $k \ge 2$ are equivalent to

$$\begin{aligned} \widetilde{\mathbf{w}}_R^\top \mathbf{u}_{nk} + \widetilde{b}_R &\ge \mathrm{logit}(\tau_R), \\ \widetilde{\mathbf{w}}_H^\top \mathbf{u}_{nk} + \widetilde{b}_H &\le \mathrm{logit}(\tau_H), \\ \widetilde{\mathbf{w}}_R^\top \mathbf{u}_{n(k-1)} + \widetilde{b}_R &\ge \mathrm{logit}(\tau_S). \end{aligned} \tag{B.5}$$

The preceding Readiness logit can be stored instead of recomputed. This equivalent implementation uses two affine scores and three comparisons at each eligible anchor, with arithmetic cost $O(d_h)$ once the states are available. The equivalence assumes thresholds strictly inside $(0, 1)$ and does not require numerical clipping.

B.3 Label Relations and Learned Scores

The labels in Equation 1 satisfy $\ell^R + \ell^H \le 1$, although ℓ^H need not equal $1 - \ell^R$. For the same distribution of anchor features $\mathbf{Z}$, let $\eta_R(\mathbf{z}) = \Pr(\ell^R = 1 \mid \mathbf{Z} = \mathbf{z})$ and $\eta_H(\mathbf{z}) = \Pr(\ell^H = 1 \mid \mathbf{Z} = \mathbf{z})$. Taking conditional expectations gives

$$\eta_R(\mathbf{z}) + \eta_H(\mathbf{z}) \le 1. \tag{B.6}$$

The relation holds under the common problem-equal weighting in Equation 4 without an independence assumption. Consequently, for the true conditional probabilities,

$$\tau_R + \tau_H \geq 1, \quad \eta_R(\mathbf{z}) \geq \tau_R \quad \Longrightarrow \quad \eta_H(\mathbf{z}) \leq 1 - \tau_R \leq \tau_H. \tag{B.7}$$

SteadyStop instead fits two separate linear sigmoid scores without imposing Equation B.6. Hazard can therefore change the fitted stopping rule. In this threshold region, a Hazard veto after the other conditions pass implies $R_{nk} + H_{nk} > \tau_R + \tau_H \geq 1$: the fitted scores depart from the true probability constraint. The gate ablations evaluate this fitted decision rule; they do not by themselves identify the source of its gains.

C Complete Case Study and Analysis

This appendix presents a complete AMC example, including the shared prefix, the supplied finalization suffix, the SteadyStop continuation, and the uninterrupted raw continuation. Raw and SteadyStop generate 2,738 and 1,301 tokens, respectively, including their final answers. These are single-case output lengths, not trigger positions.

Problem. m/n is the irreducible fraction value of

$$3 + \frac{1}{3 + \frac{1}{3+\frac{1}{3}}},$$

what is the value of $m + n$?

The shared prefix is shown once, followed by the two alternative continuations. All model text is reproduced verbatim, including repetitions, disfluencies, and literal Markdown; no spelling, grammar, or formatting has been normalized. Box titles and the trigger annotation are editorial, and supplied prompt text is distinguished from generated output.

Shared prefix (both sequences)

```
<think>
Okay, so I need to find the irreducible fraction value of this
expression: 3 + 1/(3 + 1/(3 + 1/3)), and then find m + n where m/n is
that fraction. Alright, let me start by understanding the expression
step by step.

First, let me write down the expression clearly:

3 + 1/(3 + 1/(3 + 1/3))

Hmm, that's a nested fraction. It looks like it's going to require
some simplification from the inside out. Let me try to compute the
innermost part first.

The innermost part is 3 + 1/3. Let me calculate that. 3 is the same
as 9/3, so adding 1/3 gives 10/3. So, 3 + 1/3 = 10/3. Okay, so that
part is 10/3.

Now, moving one level up, the next expression is 3 + 1/(10/3). Let me
compute that. Dividing by a fraction is the same as multiplying by
its reciprocal. So, 1 divided by 10/3 is 3/10. Therefore, the
expression becomes 3 + 3/10. Let me convert 3 to tenths to add them:
3 is 30/10, so 30/10 + 3/10 = 33/10. So that part is 33/10.

Now, moving one more level up, the entire original expression is 3 +
1/(previous result). Wait, the previous result was 33/10. So, the
expression is 3 + 1/(33/10). Let me compute that. Again, dividing by
a fraction is multiplying by its reciprocal. So 1 divided by 33/10 is
10/33. Therefore, the entire expression becomes 3 + 10/33.

Again, converting 3 to 33/11? Wait, no. Let me do it properly. 3 is
99/33, right? Because 3 * 33 = 99. So 99/33 + 10/33 = 109/33. So the
entire expression simplifies to 109/33.

Wait, but let me check that again step by step to make sure I didn't
make a mistake.

Starting again:

Start with the innermost fraction:

3 + 1/3 = 10/3. Correct.
```

Shared prefix (both sequences) (continued)

```
Then, the next level is 3 + 1/(10/3). Let me compute 1/(10/3) = 3/10.
Then 3 + 3/10 = 33/10. Correct.

Then, the next level is 3 + 1/(33/10). So 1 divided by 33/10 is
10/33. Then 3 + 10/33. 3 is 99/33, so 99/33 + 10/33 = 109/33. So
that's the result.

Now, I need to check if 109/33 is in its irreducible form. To check
that, I need to see if 109 and 33 have any common factors.

First, let's factor 33. 33 is 3 * 11. Let me check if 109 is
divisible by 3 or 11.

Dividing 109 by 3: 3*36 = 108, so 109 - 108 = 1. So remainder 1. Not
divisible by 3.

Dividing 109 by 11: 11*9 = 99, 109 - 99 = 10. Not divisible by 11.
Therefore, 109 and 33 have no common factors besides 1. Therefore,
109/33 is irreducible.

Therefore, m is 109 and n is 33, so m + n is 109 + 33 = 142.

Wait, but let me check again. Maybe I made a mistake in the
calculation steps?

Let me verify each step once more.

First, innermost part: 3 + 1/3. 3 is 3, 1/3 is 0.333..., so 3 +
0.333... is 3.333..., which is 10/3. Correct.

Next, 3 + 1/(10/3). 1/(10/3) is 3/10. Then 3 + 3/10 is 3.3, which is
33/10. Correct.

Then, 3 + 1/(33/10). 1/(33/10) is 10/33. Then 3 + 10/33. 3 is 99/33,
so 99/33 + 10/33 = 109/33. Correct.

So 109/33 is indeed the value. Since 109 is a prime number? Wait, is
109 prime?

Let me check. 109 divided by 2? No, it's odd. Divided by 3? 3*36 =
108, 109-108=1. Not divisible by 3. Divided by 5? Ends with 9, so no.
7? 7*15=105, 105 +7=112, so no. 11? 11*9=99, 11*10=110, so no.
```

Trigger boundary and SteadyStop branch. The trigger follows the final period of `11? 11*9=99, 11*10=110, so no.` above, while the thinking region is still open. SteadyStop supplies the suffix below; the model's continuation completes its open brace directly, producing `Final Answer: \boxed{142}`.

SteadyStop supplied suffix (not generated)

```
</think>
Final Answer: \boxed{
```

SteadyStop generated continuation

```
142}
```

Raw branch. Without intervention, the shared prefix is followed by the complete continuation below. It includes further checks, alternative derivations, and the full final response; no text is omitted.

Raw continuation (after the shared prefix)

```
13? 13*8=104, 13*9=117. So no. So 109 is prime. Therefore, 109 and 33
share no common factors. Therefore, the fraction is irreducible.

Therefore, m is 109 and n is 33, so m + n is 142.

But wait, just to be thorough, let me check if I interpreted the
original expression correctly. The original expression is 3 + 1/(3 +
1/(3 + 1/3)). So, the innermost is 3 + 1/3, then the next is 3 + 1
over that, then the next is 3 + 1 over that. So, yes, that's how I
did it. So the steps are correct.

Alternatively, maybe I can approach this problem by setting up
equations. Let me try that.

Let me denote the entire expression as x. So:

x = 3 + 1/(3 + 1/(3 + 1/3))

Let me compute the innermost part first, which is 3 + 1/3 = 10/3.
Then, let me denote y = 3 + 1/(10/3) = 3 + 3/10 = 33/10. Then, x = 3 +
1/y = 3 + 1/(33/10) = 3 + 10/33 = 109/33. Same result. So that's
consistent.

Alternatively, maybe I can write the entire expression as a continued
fraction and solve it algebraically. Let me try that approach.

Let me let x = 3 + 1/(3 + 1/(3 + 1/3))

Let me denote the innermost part as A = 3 + 1/3 = 10/3.

Then, the next level is B = 3 + 1/A = 3 + 1/(10/3) = 3 + 3/10 =
33/10.

Then, x = 3 + 1/B = 3 + 1/(33/10) = 3 + 10/33 = 109/33. Same result.

Alternatively, if I wanted to write this as an equation, maybe. Let
me think.

But since the expression is already broken down step by step, and
each step is straightforward, I think my answer is correct.

Wait, but just to be 100% sure, let me compute decimal approximations
to check.

Original expression: 3 + 1/(3 + 1/(3 + 1/3))

Compute innermost: 3 + 1/3 approximately 3.333333...

Then, 3 + 1/(3.333333...) approximately 3 + 1/3.333333...
approximately 3 + 0.3 approximately 3.3. Wait, 1 divided by 3.333...
is 0.3. So 3 + 0.3 is 3.3, which is 33/10. Then, 3 + 1/(33/10) is 3 +
10/33 approximately 3 + 0.303030... approximately 3.303030...

Now, 109 divided by 33 is approximately 3.303030..., which is
correct. So decimal approximation checks out.

So 109/33 is approximately 3.303, which is the value of the original
expression. So that seems correct.
```

Raw continuation (after the shared prefix) (continued)

```
Therefore, the fraction is 109/33, which is irreducible, so m + n is
109 + 33 = 142.

Wait, but let me check once more if 109 and 33 have any common
factors. 33 is 3*11. 109 divided by 3 is 36.333... Not integer. 109
divided by 11 is 9.909... Not integer. So yes, no common factors.
Therefore, 109/33 is reduced.

Therefore, the answer is 109 + 33 = 142.

**Final Answer**
\boxed{142}
</think>

To solve the expression:

$$
3 + \frac{1}{3 + \frac{1}{3 + \frac{1}{3}}}
$$

we will simplify it step by step from the innermost fraction outward.

---

### Step 1: Simplify the innermost fraction

$$
3 + \frac{1}{3} = \frac{9}{3} + \frac{1}{3} = \frac{10}{3}
$$

---

### Step 2: Simplify the next level

Now we evaluate the expression:

$$
3 + \frac{1}{\frac{10}{3}} = 3 + \frac{3}{10}
$$

Convert 3 to a fraction with denominator 10:

$$
3 = \frac{30}{10} \Rightarrow \frac{30}{10} + \frac{3}{10} =
\frac{33}{10}
$$

---

### Step 3: Simplify the outermost level

Now we compute:

$$
3 + \frac{1}{\frac{33}{10}} = 3 + \frac{10}{33}
$$

Convert 3 to a fraction with denominator 33:

$$
3 = \frac{99}{33} \Rightarrow \frac{99}{33} + \frac{10}{33} =
\frac{109}{33}
$$
```

Raw continuation (after the shared prefix) (continued)

```
---

### Step 4: Check if the fraction is irreducible

We check if the numerator and denominator have a common factor:

- $ \text{GCD}(109, 33) = 1 $, since 109 is a prime number and not
divisible by 3 or 11.

Thus, the fraction $ \frac{109}{33} $ is already in its
**irreducible** form.

---

### Final Step: Compute $ m + n $

Here, $ m = 109 $, $ n = 33 $, so:

$$
m + n = 109 + 33 = \boxed{142}
$$<|im_end|>
```

C.1 Case Analysis

The shared prefix has already completed the substantive work before the trigger. It evaluates the continued fraction as $109/33$, verifies that neither 3 nor 11 divides 109, and explicitly states $m+n = 142$. SteadyStop does not stop at that first statement. It waits until a later sentence boundary, after an additional pass over the arithmetic and divisibility checks, where current Readiness, Hazard, and preceding Readiness jointly satisfy the calibrated gate.

The two branches then differ only in how they use the established solution. The SteadyStop branch closes the open thinking region, supplies the fixed answer lead-in, and generates `142}`. The raw branch first checks another possible divisor, restates the answer, reconstructs the calculation twice, verifies it with decimal approximations, checks coprimality again, and finally emits a second formatted solution. These additions do not change the returned answer.

This example illustrates the intended behavior rather than a guarantee for every trajectory. The trigger removes 1,437 of 2,738 generated tokens (52.48%) while preserving the correct answer, but it occurs only after repeated evidence that the prefix can support finalization. The saving therefore comes from truncating redundant verification and re-expression, not from guessing at the first plausible intermediate result.

D Detailed Transfer Results

Table 5 expands the transfer results in Section 4.5 with mean generated output lengths for full reasoning and SteadyStop. Evaluation covers all 1,221 CommonsenseQA validation examples and all 1,172 ARC-Challenge test examples. Each controller is fitted and calibrated only on mathematics and reused without task-specific retraining or threshold changes. Decoding settings, three-run aggregation, and token accounting follow Section 4.1.

Method	CommonsenseQA			ARC-Challenge		
	Acc.↑	Tokens↓	Red.↑	Acc.↑	Tokens↓	Red.↑
Qwen3-1.7B						
Full reasoning	73.46	1,294.86	0.00	88.48	1,353.86	0.00
SteadyStop	70.93	727.44	43.82	85.67	719.61	46.85
DeepSeek-R1-Distill-Qwen-14B						
Full reasoning	81.33	596.43	0.00	94.88	610.91	0.00
SteadyStop	81.08	512.63	14.05	93.94	522.69	14.44

Table 5: Detailed transfer results (three-run means). Acc. and Red. are percentages, with Red. measured relative to full reasoning; Tokens is mean generated output length. Controllers are unchanged from the mathematical experiments.