# Compute When Worth It: Risk Control for Reasoning on a Compute Budget

**Anonymous Authors**[1]

## Abstract

Reasoning Large Language Models (LLMs) enable test-time scaling, with dataset-level accuracy improving as the token budget increases, motivating *adaptive* reasoning—spending tokens when they improve reliability and stopping early when additional computation is unlikely to help. However, setting the token budget, as well as the threshold for adaptive reasoning, is a practical challenge that entails a fundamental risk-accuracy trade-off. We re-frame the budget setting problem as risk control, limiting the error rate while minimizing compute. Our framework introduces an *upper* threshold that stops reasoning when the model is confident (risking incorrect output) and a novel parametric *lower* threshold that preemptively stops unsolvable instances (risking premature stoppage). Given a target risk and a validation set, we use distribution-free risk control to optimally specify these stopping mechanisms. For scenarios with multiple budget controlling criteria, we incorporate an efficiency loss to select the most computationally efficient exiting mechanism. Empirical results across diverse reasoning tasks and models demonstrate the effectiveness of our risk control approach, demonstrating computational efficiency gains by from the lower threshold and ensemble stopping mechanisms while adhering to the user-specified risk target.

## 1. Introduction

Reasoning LLMs enable test-time scaling: Spending more reasoning tokens often yields better performance (DeepSeek-AI et al., 2025; Snell et al., 2024). However, a practical challenge of reasoning LLMs lies in inducing *adaptive* reasoning behavior that adjusts to instance difficulty—deciding when additional thinking is still useful versus wasteful. Recent works propose adaptive thinking mechanisms (Wang et al., 2025a; Yang et al., 2025; Fu et al., 2025) by monitoring the reasoning model's uncertainty in the answer (e.g. by measuring confidence or entropy), and when uncertainty falls below a pre-defined threshold, the reasoning is halted. Adaptive thinking enables instance-dependent token budgets, since the reasoning effort required to reach a confident threshold varies by problem. However, adaptive thinking does not alleviate the practical challenge of setting reasoning budget; it only converts the problem of setting a token budget into setting a threshold. In fact, setting a threshold could be trickier than setting a token budget since the threshold value often lacks interpretable meaning, and could lie in arbitrary ranges depending on how the uncertainty is computed (Figure. 1, left).

In this paper, we provide a principled framework for setting stopping rules for adaptive reasoning. In particular, we leverage a key implication of test-time scaling: *any early termination of reasoning introduces a risk of error*. Based on this observation, we reframe the problem of setting a reasoning budget as choosing an acceptable level of risk, an interpretable quantity that directly interfaces with downstream decision-making pipelines. We utilize an external validation set and distribution-free risk control framework (Bates et al., 2021; Jazbec et al., 2024) to automatically map a user-specified risk to the corresponding criteria for terminating the reasoning chain.

Specifically, we delineate two complementary types of risks and their corresponding controlling mechanisms (illustrated in Fig. 2): False positive risk of thinking the model has the correct answer, controlled by an upper threshold of confidence; False negative risk of thinking the current (and future) answers will be incorrect, controlled by the lower threshold, a novel mechanism that measures whether the reasoning is making enough progress. These two types of risks and thresholds are related to different sources of inefficiency: The upper threshold reduces wasted tokens after the model has effectively converged (Eq. (8)), while the lower threshold avoids spending tokens when further reasoning is unlikely to help (Eq. (9)).

In summary, our contributions are: (i) We introduce a collec-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

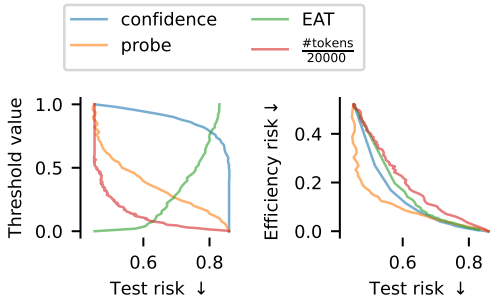



*Figure 1.* **Early-stopping behavior under different target test risks.** ***Left:*** Threshold values required to achieve a given target test risk vary substantially across early-stopping signals, indicating that threshold selection is signal-dependent. ***Right:*** The relative efficiency of different early-stopping methods depends on the target test risk, and no single method is uniformly most efficient across all risk levels. Results shown here use an upper-threshold-only stopping rule; introducing both upper and lower thresholds would further complicate the risk-threshold mapping.

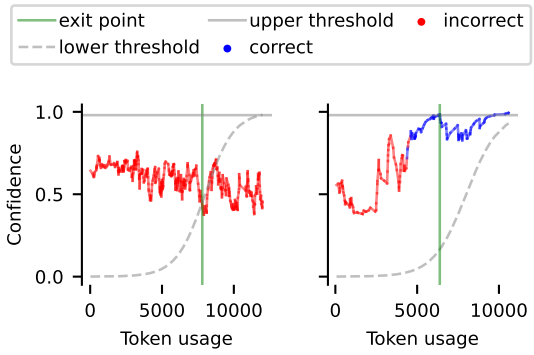



*Figure 2.* **Dual-threshold early exit via risk-controlled confidence dynamics.** We plot confidence trajectories as a function of token usage under Qwen3-8B on AIME questions. **Left:** an unsolvable instance, model confidence fluctuates and fails to reach the upper threshold; the reasoning is halted early by the *parametric lower threshold*, preventing unnecessary token consumption. **Right:** a solvable instance, where confidence steadily increases and crosses the *upper threshold*, triggering termination once sufficient confidence is achieved.

tion of loss functions that capture different sources of inefficiency and errors from early stopping, enabling incorporation of distribution-free risk control into reasoning LLMs, a principled approach that allows early stopping (Sec. 4.1); (ii) We introduce a novel parametric lower threshold that halts the reasoning when the model is not progressing sufficiently at decreasing uncertainty (Sec. 4.2); (iii) We demonstrate that at the same risk level, different approaches exhibit different efficiency, whereas using our upper threshold plus our proposed parametric lower threshold consistently yields efficiency gains (Sec. 5.3).

## 2. Background: Adaptive early stopping via upper thresholding confidence

**Reasoning models overthink.** Recent reasoning LLMs are post-trained to output an explicit reasoning trace in a delimited format, followed by a final answer:

$$y \;=\; \langle \texttt{think} \rangle \, r_{1:T} \, \langle \texttt{/think} \rangle \, a, \tag{1}$$

where a complete generation $y$ is composed of: $r_{1:T}$ the reasoning segment of length $T$ (tokens or steps) wrapped inside special beginning/end of thinking tokens: <think> and </think>, and $a$ the final answer. A common observation is that $T$ can be much larger than is necessary on many instances ("overthinking"), in turn introducing unnecessary inference cost. In particular, models often continue to reason when a correct answer can already be elicited.

**Uncertainty monitoring for adaptive early stopping.** Given an input $x$, let $r_{1:t}$ denote the partial reasoning trajectory upto $t$ steps. Recent works (Wang et al., 2025a; Yang et al., 2025) propose to adaptively early stop the reasoning by monitoring a scalar confidence/uncertainty signal computed from the partial trajectory:

$$s_t \;=\; u(x, r_{1:t}) \,, \tag{2}$$

where a large $s_t$ typically indicates higher confidence (or lower uncertainty, depending on convention). Common choices for $s_t$ are often derived from the statistical properties of tokens generated after $\langle$/think$\rangle$, such as their entropy (Wang et al., 2025a) or confidence (Yang et al., 2025). Note that raw signals can be noisy or have inconvenient ranges. Transformations such as smoothing, reciprocal, or normalization are often applied before thresholding:

$$\tilde{s}_t \;=\; g(s_{1:t}) \,. \tag{3}$$

Here $g(\cdot)$ may depend on the history $s_{1:t}$ (e.g., an exponential moving average) to reduce variance.

A canonical early exit policy then halts reasoning at the earliest time the transformed score *exceeds* a threshold $\lambda$:

$$\tau \;=\; \min \Big\{ t \geq 1 \;:\; \tilde{s}_t \geq \lambda \Big\}. \tag{4}$$

We will refer to this as the *upper threshold exit* mechanism in the rest of the text. The policy then emits the answer $a$ and avoids generating $r_{t>\text{exit}}$. This framing unifies a broad set of "stop-when-confident" approaches.

**Advantages of adaptive thinking.** Given a dataset, adaptive reasoning assigns a different number of tokens to each question depending on how long it takes for the confidence to reach the target threshold. This allows better allocation of the budget across instances, in contrast to assigning a fixed budget to all questions, and as a result, adaptive thinking achieves the same dataset level accuracy with fewer total tokens, i.e., more efficient reasoning.

## 3. Related Works

**Early Exit for Chain-of-Thought.** A broad literature reduces inference cost by *early exiting* chain-of-thought once additional reasoning is unlikely to help. As described in Section 2 most methods instantiate an *upper-threshold* rule: They monitor a confidence/uncertainty proxy (e.g., entropy or stability of intermediate answers) and stop when the model appears sufficiently certain or converged, via entropy signals, trial-answer confidence, or answer-consistency/convergence heuristics (Wang et al., 2025a; Yang et al., 2025; Mao et al., 2025; Liu & Wang, 2025; Liao et al., 2025; Wei et al., 2025; Fu et al., 2025; Jurayj et al., 2025).

**How We Differ:** Our framework departs from prior early-exit work in two ways. **(1) Novel lower thresholding mechanism.** Existing methods overwhelmingly address "stop when confident" (an upper threshold). We additionally formalize a *lower-confidence* stopping criterion such that we "stop when the model is confidently not making progress", and combine both into a symmetric dual-threshold rule that captures confident success *and* confident failure within a *single* trajectory. In fact, in the experiment section of Wang et al. (2025a), the authors explicitly state their limitation on challenging datasets, where most problems never reach the target upper threshold, resulting in huge token waste. **(2) Principled threshold selection with guarantees.** Prior work typically relies on hand-tuned cutoffs, sweeps, or heuristic criteria. In contrast, we provide a statistically grounded calibration procedure that selects both thresholds to satisfy user-specified accuracy/risk constraints.

**Risk control for reasoning LLM.** The closest prior work is Thought Calibration (Wu et al., 2025), which frames continuation as hypothesis testing and calibrates a threshold for the probe signal, so that stopping does not degrade accuracy, using a loss function similar to our false positive loss (Eq. (6)). Our work further introduces lower threshold mechanisms (and the corresponding loss) as well as efficiency loss into the risk control framework. PAC Reasoning (Zeng et al., 2025) aims to derive distribution-free risk guarantees for deferring a query from a thinking model to a non-thinking model through thresholding the uncertainty. Our work instead controls the reasoning process of a single model and determines when to terminate an ongoing trajectory as confident success or unsolvable failure.

## 4. Method

**Notation.** Let $y^*$ denote the ground true answer, $f_t(x)$ be the model's prediction at step $t$ given question $x$, and $T$ as the total reasoning steps. At each step $t$, the model also emits an auxiliary scalar signal $s_t \in \mathbb{R}$ (e.g., confidence), which is used to determine early stopping.

**Motivation and Goals.** Our work is motivated by two primary goals: (i) develop an adaptive early-stopping mechanism that allows user-specified control of the error rate(s), and (ii) extend existing upper-threshold-only approaches to also stop when an instance is likely to be unsolvable within $T$ reasoning steps (our maximum budget). Regarding (i), for some loss function $\ell\left(y^*, f_t(x)\right)$ that measures the quality of the intermediate solution $f_t(x)$, we wish to control the expected loss, a.k.a. risk:

$$\mathbb{E}_{x,y^*}\left[\ell\left(y^*, f_t(x); \lambda\right)\right] \;\leq\; \epsilon, \text{ where } \epsilon \in [0,1]. \tag{5}$$

The $\epsilon$ parameter, i.e., risk tolerance, is specified by the user—for example, as we shall consider, the rate at which early-terminated solutions are wrong (false positives)—and $\lambda \in [0,1]$ is the parameter of the exiting mechanism that we will tune to control the loss. As mentioned in Section 2, one way to specify $\lambda$ is by means of a confidence threshold. Yet, this brings us to motivation (ii): upper-threshold approaches aim to stop when the reasoning chain has arrived at a correct solution. Yet, given the challenging tasks to which LLMs are applied (e.g. Humanity's Last Exam, cutting-edge mathematics), we also want the reasoning to terminate when the task is too hard, meaning the LLM is unlikely to arrive at the correct answer before $T$ steps of reasoning. We now detail our construction that achieves (i) and (ii), first defining the loss functions we consider (4.1), then our dual threshold approach (4.2), and lastly our algorithms for finding the best $\lambda$, the parameters that control the thresholds, such that the risk bound is guaranteed (4.3).

### 4.1. Risk of early exiting reasoning

We begin by defining four instance-wise loss functions that capture the correctness-efficiency tradeoff induced by early stopping. For each type of loss, there is a corresponding *risk*, computed as an empirical average over all instances.

**Correctness loss.** These measure the potential error incurred by premature stopping:

- **False positive Loss:**

$$\ell_{\mathrm{FP}}^{\mathrm{upper}}(y^*, f_t(x), \tilde{s}_t(x); \lambda_+) = \mathbb{I}\left[\tilde{s}_t(x) \geq \lambda_+\right] \cdot \mathbb{I}\left[f_t(x) \neq y^*\right]. \tag{6}$$

  When we believe the model has reached a correct answer, but the induced answer $f_t(x)$ is different from the correct answer $y^*$, a false positive loss is incurred. False positive instances introduce misleading predictions for downstream decision-making tasks.

- **False negative loss:**

$$\ell_{\mathrm{FN}}^{\mathrm{lower}}(y^*, f_{t:T}(x), \tilde{s}_t(x); \lambda_-) = \frac{\mathbb{I}[\tilde{s}_t(x) \leq \lambda_-]}{T - t + 1} \cdot \sum_{T \geq k \geq t} \mathbb{I}\left[f_k(x) = y^*\right]. \tag{7}$$

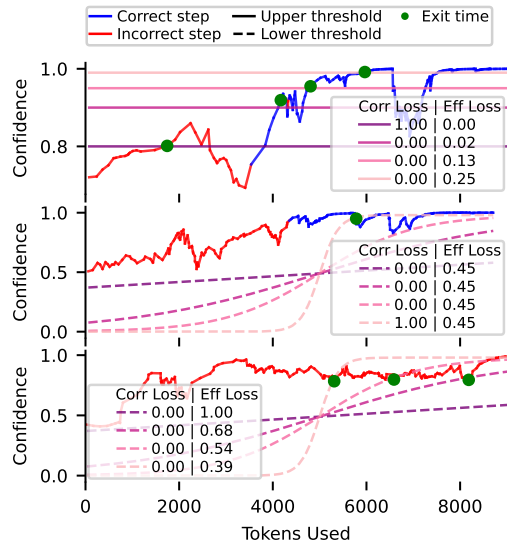



*Figure 3.* **Visualization of the proposed correctness and efficiency losses under different thresholds.** Lines of different colors (purple to pink) denote different threshold curves. Numbers in the box show the correctness and efficiency loss for each threshold. The top row shows the upper-threshold correctness and efficiency loss (Eq. (6) and (8)). Bottom two figures show lower-threshold sigmoid curves (Eq. (12)) and the corresponding losses the lower-threshold correctness loss (Eq.(7) and (9)). For a given instance, when multiple thresholds have the same correctness loss (e.g. top 3 lines in first row, all lines in the bottom row), the one with smallest efficiency loss is preferred, in order to maximize efficiency while maintaining user provided risk tolerance.

When we believe the model cannot solve the problem, halt the reasoning early and defer the instance to experts, but the model would reach the correct answer given additional reasoning, we incur a false negative loss. False negative instances reduce the usability of the model and introduce cost for calling external resources. Importantly, Eq. (7) is computed as the correctness averaged over all future steps; therefore, if there only exists very sparse correct future steps (e.g. through random guessing), Eq. (7) will be small. This contrasts Eq. (6) with using the *existence* of correct future steps as the loss, which incurs a loss of 1 even if, e.g. it is just one step out of 1,000 future steps.

**Efficiency Loss.** Now we define two losses that measure the efficiency of early stopping at step $t$

- **Upper threshold efficiency loss:**

$$\ell_{\text{eff}}^{\text{upper}}(t) = \frac{1}{T}\max(0, t - t'), \tag{8}$$

where $t' = \min\{t \le T : f_t(x) = y^*\}$ denotes the first step at which the model produces a correct answer and $T$ denotes the total reasoning budget. When a model arrives at a correct answer at step $t'$, any additional reasoning is wasted computation. This loss measures normalized regret: the fraction of total steps spent deliberating after the answer was already correct.

- **Lower threshold efficiency loss**

$$\ell_{\text{eff}}^{\text{lower}}(t) = \frac{1}{T}\sum_{k \le t} \mathbb{I}[y^* \neq f_k(x)]. \tag{9}$$

We want to know how much of the budget we have wasted so far on unsolvable questions. A low value suggests that either we stopped very early or the model solved the problem. A high value, (in the extreme case, 1), suggests that we use lots of budgets without making much progress.

### 4.2. Early stopping with two thresholding mechanisms.

Having defined the relevant correctness and efficiency risks in Section 4.1, we now describe how these risks are applied through two complementary threshold mechanisms that stop the reasoning process. Concretely, at each reasoning step $t$, the model produces an auxiliary scalar signal $s_t$. The stopping policy monitors this signal and triggers an exit once either threshold condition is met.

**Upper threshold: stopping on confidence.** The upper threshold mechanism controls the *false positive risk* $R_{\text{FP}}^{\text{upper}}$. It halts reasoning as soon as the confidence signal exceeds a preset threshold $\lambda_+$:

$$\tau_+ = \min\{t : \tilde{s}_t(x) \ge \lambda_+\}. \tag{10}$$

Intuitively, this corresponds to the model being ready to stop because it believes the current answer is correct. However, if this threshold is misplaced, early stopping leads to an incorrect prediction, incurring a false positive error. The threshold $\lambda_+$, therefore, directly trades off early exits against the risk of halting on an incorrect answer. In addition to correctness, the upper threshold affects efficiency through the regret term in Eq. (8), controlling unnecessary reasoning beyond the first correct step. This mechanism is identical to the standard upper-threshold setting discussed in Section 4.1. An example of loss v.s. threshold values is illustrated in Fig. 3, top row.

**Lower threshold: stopping on pessimism.** The lower threshold mechanism controls the *false negative loss*. Given a total reasoning budget of $B$ tokens, and $\omega_t$ the total tokens generated up to reasoning step $t$, it halts reasoning when the progress signal falls below a *parametric* threshold $\lambda_-(t; c)$:

$$\tau_- = \min\{t : \tilde{s}_t(x) < \lambda_-(t; c)\}, \tag{11}$$

$$\text{where } \lambda_-(t; c) = \sigma\big(c\,(\omega_t - \tfrac{B}{2})\big), \sigma(z) = \frac{1}{1+e^{-z}} \tag{12}$$

where $c$ controls the shape of the lower threshold function. Different from the upper threshold mechanism, we design the lower threshold as a parametric function (rather than a

**Algorithm 1** Finding threshold given $\epsilon$ and validation set

**Require:** Validation set $\mathcal{V} = \{(x_i, y_i)\}_{i=1}^n$
**Require:** Risk budget $\epsilon \in [0, 1]$
**Require:** Candidate uncertainty signal set $\mathcal{S}$
**Require:** Threshold grids $\{\Lambda_s\}_{s\in\mathcal{S}}$
**Require:** Prediction procedure that yields a model output $\hat{y}_i$ for each $x_i$
**Require:** Risk estimator $\widetilde{\text{Risk}}(\mathcal{V}, s, \lambda)$
**Require:** Efficiency loss estimator $\widehat{\text{Loss}}_{\text{eff}}(\mathcal{V}, s, \lambda)$
**Ensure:** Selected signal–threshold pair $(s^\star, \lambda^\star)$

**Initialize feasible set:** $\mathcal{C} \leftarrow \emptyset$
**for** each signal $s \in \mathcal{S}$ **do** ▷*Grid search over signals*
  **for** each threshold $\lambda \in \Lambda_s$ **do** ▷*and thresholds*
    Compute adjusted estimated risk: $r \leftarrow \widetilde{\text{Risk}}(\mathcal{V}, s, \lambda)$
    **if** $r \le \epsilon$ **then**
      Compute efficiency loss: $\ell \leftarrow \widehat{\text{Loss}}_{\text{eff}}(\mathcal{V}, s, \lambda)$
      Add $(s, \lambda, \ell)$ to $\mathcal{C}$
**if** $\mathcal{C} = \emptyset$ **then** ▷*No feasible pair*
  **return** (NONE, NONE)
**else**
  $(s^\star, \lambda^\star) \leftarrow \arg\min_{(s,\lambda,\ell)\in\mathcal{C}} \ell$ ▷*Select optimal pair.*
  **return** $(s^\star, \lambda^\star)$

fixed line) that increases with token consumption, reflecting the intuition that model confidence should gradually increase as reasoning proceeds, with the rate controlled by $c$, and if not, we may conjecture that the model may never solve the problem. Of course, the lower threshold could cause *false pessimism*: The signal suggests that continued reasoning is unnecessary due to slow progress, even though a correct answer is reachable at a later step. Lower threshold also affects the efficiency loss defined in Eq. (9), which quantifies wasted computation on problems that remain unsolved despite extended reasoning. An ideal lower threshold should prevent excessive token usage on inherently unsolvable instances while minimizing premature exits on solvable ones. Examples of loss v.s. threshold shapes are presented in Fig. 3, bottom two rows.

### 4.3. Picking threshold values using validation set (Alg. 1)

Now that we have defined the early stopping mechanisms and their corresponding risks, the goal becomes converting a user-specified risk tolerance $\epsilon \in [0, 1]$ into an early stopping or abstention rule, in particular, the upper threshold value $\lambda_+$ and the parametric function's parameter $c$, whose induced risk does not violate users' specified values. Note that in this section, we drop the superscript *lower / upper* and the subscript $+/-$ for notation simplicity; the procedure described below applies to both types of stopping mechanisms.

To find the threshold parameter, we utilize a labeled validation set

$$\mathcal{V} = \{(x_i, y_i^*)\}_{i=1}^n \tag{13}$$

Then for each signal $s \in \mathcal{S}$ and each threshold (or value of $c$) $\lambda \in \Lambda_s$, we estimate the induced risk

$$\widehat{\text{Risk}}(\mathcal{V}, s, \lambda) = \frac{1}{n}\sum_{i=1}^n \ell(y_i^*, f_\tau(x_i), \tilde{s}_\tau(x_i); \lambda), \tag{14}$$

where $\tau$ denotes the exit position induced by signal $s$ under threshold $\lambda$. *Naively*, we can keep the pairs whose estimated risk on the validation set does not exceed the user budget $\epsilon$, However, since $\mathcal{V}$ is finite (and often small), the validation risk can be an optimistic estimate of the true risk; as a result, thresholds selected by naive cross-validation can *overfit* to the validation set and violate the target risk on unseen test data. The first column in Figure 4 illustrates this phenomenon: the naive calibration frequently yields realized test risk above the target line $y = x$.

To mitigate finite-sample error, we replace the naive feasibility check with a **distribution-free risk control** criterion that adds a margin to account for validation uncertainty (see Appendix B for details). In implementation, this corresponds to replacing the raw empirical risk with an *adjusted* risk,

$$\widetilde{\text{Risk}}(\mathcal{V}, s, \lambda) = \widehat{\text{Risk}}(\mathcal{V}, s, \lambda) + (\text{finite-sample correction}), \tag{15}$$

where (finite-sample correction) is a function of the validation set size, and we only keep pairs where $\widetilde{\text{Risk}}(\mathcal{V}, s, \lambda) \le \epsilon$. Intuitively, this makes the feasibility test more conservative when $n$ is small, reducing the chance that the selected threshold underestimates risk. The second column in Figure. 4 illustrates the effect of finite sample correction, where the test risk now consistently stays below the user-specified risk tolerance across all random trials.

In case multiple feasible pairs of $(s, \lambda)$ exist (for example, different types of uncertainty signal and thresholds), we select the one that minimizes an *efficiency loss*,

$$\widehat{\text{Loss}}_{\text{eff}}(\mathcal{V}, s, \lambda) = \frac{1}{n}\sum_{i=1}^n \ell_{\text{eff}}\left(\tau^i(s, \lambda)\right), \tag{16}$$

where $\tau^i(s, \lambda)$ denotes the exit position for the $i$th validation sample under signal $s$ and threshold parameter $\lambda$. This routine picks the most efficient signal and threshold pair in $\mathcal{S} \times \Lambda$ that do not exeed the pre-specified risk $\epsilon$.

**Putting it together.** Our final selection rule is: (i) enumerate signal - threshold candidates on $\mathcal{V}$, (ii) enforce the risk budget using risk control to account for finite-sample error, and (iii) among all feasible candidates, choose the one that minimizes the efficiency loss estimate. This yields a simple, plug-in calibration interface for arbitrary signals while providing stronger protection against validation overfitting than naive threshold tuning.

### 4.4. Combining thresholds

In deployments, a single stopping criterion is insufficient: We may want to use lower and upper thresholds jointly. To

achieve that, we consider a two-step approach, where we assume the user would provide two risk tolerances, $\epsilon_+$ and $\epsilon_-$, which target at Eq. (6) and Eq. (7) respectively. Now given a signal $s$, we first search for the $\lambda_+$ that satisfies $\epsilon_+$; Then we search for (parameters of) $\lambda_-$ that satisfy $\epsilon_-$, where we would upper bound the sigmoid curve by $\lambda_+$ through multiplying it by $\lambda_+$, such that the lower and upper thresholds never intersect.

However, the risk control guarantee would only partially hold in this case on the test set: Since the lower threshold is introduced, the false positive rate is now computed as the ratio of incorrect predictions *in all un-abstained instances*, which may increase or decrease compared with the test risk without abstention, a quantity controlled by $\epsilon$. In practice, if the uncertainty signals are well separated for solvable and unsolvable questions, such scenarios are unlikely to happen. The false negative risk, on the other hand, will not be affected, as it is already chosen conditioned on the existence of the upper threshold. To resolve this issue, one could consider jointly searching a product space of $\lambda_+$ and $\lambda_-$, which have stronger guarantees but would incur a significant overhead at the calibration phase.

At inference time, given an instance, let $\tau_+$ and $\tau_-$ be the exit times induced by the upper and lower thresholds, we would simply exit at

$$\tau \;=\; \min\{\tau_+, \tau_-, T\}. \tag{17}$$

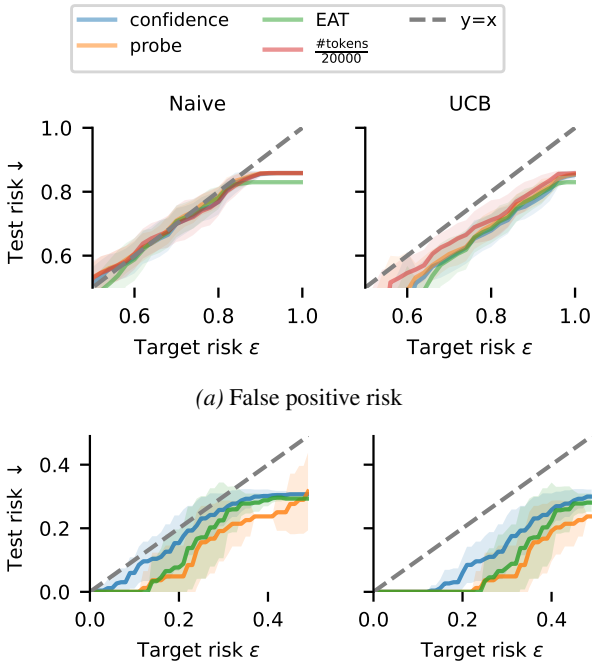
where $T$ denotes the maximum token budget, and is triggered if neither threshold is hit.

### 4.5. Advantages of the risk-control perspective

Note that there are two settings where risk control is NOT strictly necessary.

- **Purely comparative evaluation.** If the goal is only to demonstrate that one early stopping signal dominates another in the accuracy–compute trade-off, then a threshold sweep is sufficient: one can enumerate thresholds and report the resulting Pareto frontier (or area-under-curve) on a labeled benchmark, without committing to any particular operating point.
- **Perfectly interpretable signals.** If the stopping signal is directly calibrated to correctness, e.g., a confidence value $p$ truly means $\Pr(\text{correct}) = p$, then choosing a threshold is equivalent to choosing an error rate, and ad-hoc thresholding becomes less problematic.

**Why these conditions are atypical in practice.** The first scenario diverges from deployment settings in two ways. First, real systems typically require *one* concrete configuration (or a small set of configurations) that meets a user- or application-specified error tolerance, rather than reporting an entire sweep curve post hoc. The second condition also rarely holds for existing early-stopping signals: The mapping from a raw threshold to the achieved error is highly signal-, model-, and task-dependent (Fig. 1, left). Moreover, the *lower* threshold, which is often implemented as a parametric function of time/tokens, contains more hyperparameters that are even harder to interpret and transfer across tasks, making hand-tuning unreliable and amplifying the need for principled, finite-sample-aware selection.




*(a)* False positive risk




*(b)* False negative risk

*Figure 4.* **Empirical verification of risk control.** We plot the empirical test risk (y-axis) against the user-specified target risk $\epsilon$ (x-axis). Solid lines and shaded regions indicate the mean and standard deviation over 40 random test-validation splits. Different colors denote different early-stopping signals. The left panel (NAIVE) selects thresholds on the validation set without finite-sample correction, leading to frequent violations in the realized test risk exceeding $\epsilon$, particularly on false negative risk (Eq. (7)) controlled by the lower threshold (Eq. (12)), which has more flexibility and therefore more prone to noise. The right panel (UCB) applies a probabilistic risk control procedure that accounts for validation uncertainty, guaranteeing that the test risk under the selected threshold is upper-bounded by $\varepsilon$ with high probability.

**What risk control buys us.** Our framework therefore, asks the user to specify a *risk tolerance* $\epsilon$, an operational quantity with downstream meaning, and uses a held-out validation set with finite-sample correction to automatically select thresholds (and, when applicable, choose among multiple stopping criteria) so that the realized test risk respects the user-specified tolerance with high probability. This shifts the burden from tuning opaque thresholds to selecting an interpretable error tolerance.

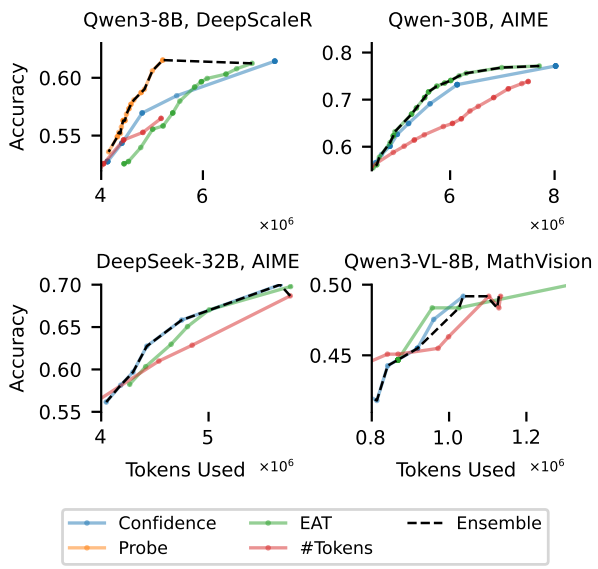



*Figure 5.* **Ensemble of signals improves efficiency.** Under four models, we consider upper-threshold only early stopping. Given a target tolerance $\epsilon$, risk control framework picks the signal that minimizes the efficiency loss (Eq. (8)), forming an *ensemble* of signals, which translates to superior efficiency on the test set (better accuracy v.s. token trade-off).

## 5. Empirical validation

In this section, we provide empirical verification of the effectiveness of risk control. Then we demonstrate the efficiency gain from the risk control framework: From ensembling signals and from combining upper and lower thresholds.

### 5.1. Experiment setup

**Models and datasets** We evaluate our methods on Qwen3-8B, Qwen3-30B-A3B, DeepSeek-R1-Distill-Qwen-32B, as well as Qwen3-VL-8B. For datasets, we consider AIME (1983-2025, 1011 samples), a subset of DeepScaleR (Luo et al., 2025, 1189 samples) with AIME removed, GPQA-Diamond (Rein et al., 2024, 198 samples) and MathVision (Wang et al., 2024, 304 samples), a vision language reasoning dataset.

**Reasoning generation** We use a system prompt of "Please reason step by step, and put your final answer within \boxed{}.". For all models, we use the recommended decoding configurations from the model cards. The generation is conducted using vLLM for all experiments.

**Answer and uncertainty signal elicitation** During the generation of reasoning, we consider text between two consecutive `\n\n` delimiters as a chunk; all uncertainty signals are computed at the end of each growing chunk. If we decide to early stop at a chunk, we append the forcing string `"\n **Final Answer**\n \boxed{"` to elicit a canonical boxed answer. We consider 2 uncertainty-based signals: Confidence (Yang et al., 2025) and EAT (Wang et al., 2025a); On Qwen3-8B, we trained a probe model (Zhang et al., 2025) on AIME. Additionally, we consider using the number of tokens as a stopping criterion. Appendix A.1 provides an introduction to the signals.

### 5.2. Risk control framework controls risk

We begin by performing a sanity check on the effectiveness of Alg. 1. In particular, we aim to verify, given a risk tolerance $\epsilon$, whether the threshold picked on the validation set gives a test risk smaller than $\epsilon$.

We consider Qwen3-8B model on AIME, under a variety of early stopping signals, including uncertainty-based, probe-based, and just by the number of tokens. We randomly generate 40 validation-test splits, with a validation set size of 50 samples (5 percent), and we study both false positive (Eq. (6)) and false negative loss (Eq. (7)), We enumerate over values of $\epsilon$ between $[0, 1]$ with a step size of $0.01$, find the thresholds using Alg. 1, and then compare the achieved risk on the test set with $\epsilon$.

The results are demonstrated in Fig. 4. In particular, using finite sample correction (UCB) enables the risk on the unseen test set to lie below the user-specified threshold; Naive cross-validation, despite having the mean controlled, its standard deviation bands often cross the $y = x$ line, i.e., many individual runs exceed the target risk.

### 5.3. Risk control framework improves efficiency

Now we demonstrate how the risk control framework enables more *efficient* reasoning. The setting involves enumerating over epsilons, then for each epsilon, we check the total number of questions correctly answered by the model v.s. the total number of tokens used.

**Ensembling signal improves efficiency** Given an $\epsilon$, a critical step of Alg. 1 is to select a signal threshold pair that minimizes the efficiency loss (line 6 and 11). This enables us to construct an ensemble of uncertainty signals where we pick the most efficient signal and threshold for each $\epsilon$ and dataset. Our results confirm that the most efficient ones on the validation set transfer to efficiency gain on the test set, as is shown in Fig. 5: Overall ensemble manages to picks the most efficient signal across all candidates, e.g. on Qwen3-8B, we have access to a powerful probing model trained on AIME, so ensemble consistently picks probe across $\epsilon$.

**Lower threshold improves efficiency.** Next we examine how the *upper* and *lower* thresholds complement each other in improving efficiency. The upper threshold mainly saves tokens on *solvable* instances by stopping once the model becomes confident, whereas the lower threshold mainly saves tokens on *unsolvable* instances by halting runs whose un-

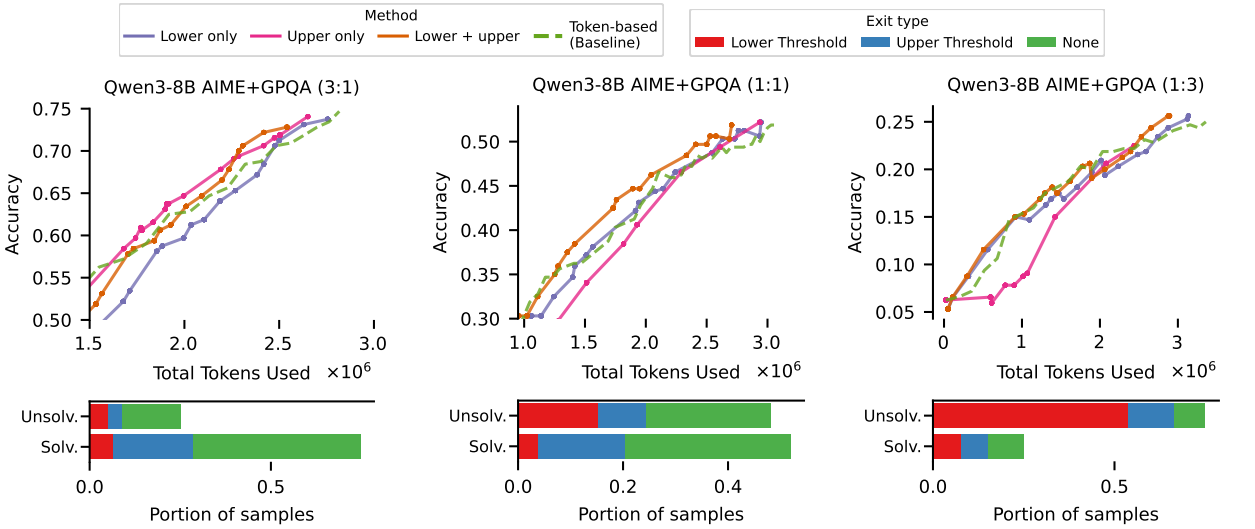



*Figure 6.* **Lower-threshold gains grow when unsolvable instances are prevalent.** We evaluate Qwen3-8B with confidence as the uncertainty signal on datasets with *solvable:unsolvable* ratios of 3:1, 1:1, and 1:3 (constructed by pooling AIME and GPQA, labeling instances by solvability under the full token budget, and subsampling to match each ratio). **Top:** test accuracy (instances abstained by lower threshold considered as wrong) versus total tokens for LOWER-ONLY, UPPER-ONLY, and LOWER+UPPER; each point corresponds to a validation-calibrated risk tolerance $\epsilon$, and we include a uniform token-budget baseline. When unsolvable instances are common (1:1 and 1:3), UPPER-ONLY clusters near the high-token regime because many runs never reach the confidence cutoff and thus run to the budget limit, while adding the lower threshold shifts the curve left (similar accuracy with fewer tokens). **Bottom:** for LOWER+UPPER at a representative operating point (second-highest accuracy), we report the solvable/unsolvable fractions and which condition triggered termination, showing that solvable instances typically exit via the upper threshold and unsolvable ones via the lower threshold.

certainty fails to improve and would otherwise consume the full budget. Thus, the value of a lower threshold depends on the *solvable:unsolvable* composition of the test set. To study this dependence, we construct evaluation sets with controlled **solvable:unsolvable ratios** of 3:1, 1:1, and 1:3. We pool AIME and GPQA, label each instance by whether the model can reach the correct final answer under the full token budget, and subsample to match the target ratio. Using confidence as the uncertainty signal, we compare UPPER-ONLY, LOWER-ONLY, and LOWER+UPPER. For UPPER-ONLY and LOWER-ONLY, each point on the curve corresponds to a risk tolerance $\epsilon$ calibrated on the validation split. For LOWER+UPPER, we fix the upper threshold to the value achieving the smallest validation risk and then sweep the lower-threshold $\epsilon$ following Sec. 4.4.

As shown in Fig. 6, when solvable instances dominate (3:1), UPPER-ONLY captures most savings and LOWER-ONLY adds little. When unsolvable instances are common (1:1 and 1:3), UPPER-ONLY becomes inefficient: its curve stays near the high-token region because many runs never satisfy the confidence cutoff and therefore run to the maximum budget (Fig. 6, top row). Adding the lower threshold mitigates this failure mode and yields a clear leftward shift at comparable accuracy. The bottom row of Fig. 6 confirms this division of labor: at a representative operating point (second-highest accuracy for LOWER+UPPER), solvable instances mostly exit via the upper threshold, while unsolvable instances mostly exit via the lower threshold.

### 5.4. Ablation study

Lastly, we study the robustness of risk control under variation of validation set sizes as well as under distribution shift. We focus the study on Qwen3-8B evaluated on DeepScaleR.

**Size of validation set** We fix the test set size as 800 samples and vary the validation set size in $\{8, 16, 40\}$. Broadly, we find that as validation set size decreases, the advantage of UCB over Naive becomes more pronounced (Fig. 7).

**Distribution shift** We study two types of distribution shift between the validation and test set:

- **Length shift**: The validation set has an overall length different than the test set. Results are shown in Fig. 8: Short (validation) to long (test) brings challenges, particularly for the lower threshold risk, whose shape depends heavily on the reasoning length. The false positive risk controlled by the upper threshold does not show much degradation.
- **Dataset shift**: The validation set and the test set come from different datasets. We consider AIME v.s. GPQA-Diamond, i.e. Math v.s. Science questions. Note that this also includes difficulty shift, as GPQA has a much lower solved rate than AIME. Results are shown in Fig. 9. Again, without finite sample correction, Naive shows severe excess risk while UCB continues to bound the test risk below the target risk $\epsilon$.

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References


Bates et al., S. Distribution-free, risk-controlling prediction sets. *Journal of the ACM*, 2021.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Fu, Y., Chen, J., Zhu, S., Fu, Z., Dai, Z., Zhuang, Y., Ma, Y., Qiao, A., Rosing, T., Stoica, I., and Zhang, H. Efficiently scaling llm reasoning with certaindex, 2025. URL https://arxiv.org/abs/2412.20993.

Jazbec, M., Timans, A., Hadži Veljković, T., Sakmann, K., Zhang, D., Andersson Naesseth, C., and Nalisnick, E. Fast yet safe: Early-exiting with risk control. *Advances in Neural Information Processing Systems*, 37:129825–129854, 2024.

Jurayj, W., Cheng, J., and Van Durme, B. Is that your final answer? test-time scaling improves selective question answering. *arXiv preprint arXiv:2502.13962*, 2025.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Liao, B., Dong, H., Xu, Y., Sahoo, D., Monz, C., Li, J., and Xiong, C. Fractured chain-of-thought reasoning, 2025. URL https://arxiv.org/abs/2505.12992.

Liu, X. and Wang, L. Answer convergence as a signal for early stopping in reasoning, 2025. URL https://arxiv.org/abs/2506.02536.

Luo, M., Tan, S., Wong, J., Shi, X., Tang, W. Y., Roongta, M., Cai, C., Luo, J., Li, L. E., Popa, R. A., and Stoica, I. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Mode 2025. Notion Blog.

Mao, M., Yin, B., Zhu, Y., and Fang, X. Early stopping chain-of-thoughts in large language models, 2025. URL https://arxiv.org/abs/2509.14004.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Wang, K., Pan, J., Shi, W., Lu, Z., Ren, H., Zhou, A., Zhan, M., and Li, H. Measuring multimodal mathematical reasoning with math-vision dataset. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=QWTCcxMpPA.

Wang, X., McInerney, J., Wang, L., and Kallus, N. Entropy after ⟨/Think⟩ for reasoning model early exiting, 2025a. URL https://arxiv.org/abs/2509.26522.

Wang, Y., Zhang, Y., Yu, T., Xu, C., Zhang, F., and Lian, F. Adaptive deep reasoning: Triggering deep thinking when needed, 2025b. URL https://arxiv.org/abs/2505.20101.

Wei, Z., Pang, L., Liu, J., Deng, J., Xu, S., Duan, Z., Wang, J., Sun, F., Cai, X., Shen, H., and Cheng, X. Stop spinning wheels: Mitigating llm overthinking via mining patterns for early reasoning exit, 2025. URL https://arxiv.org/abs/2508.17627.

Wu, M., Zhou, C., Bates, S., and Jaakkola, T. Thought calibration: Efficient and confident test-time scaling. *ArXiv*, abs/2505.18404, 2025. URL https://api.semanticscholar.org/CorpusID:278905640.

Yang, C., Si, Q., Duan, Y., Zhu, Z., Zhu, C., Li, Q., Lin, Z., Cao, L., and Wang, W. Dynamic early exit in reasoning models, 2025. URL https://arxiv.org/abs/2504.15895.

Zeng, H., Huang, J., Jing, B., Wei, H., and An, B. Pac reasoning: Controlling the performance loss for efficient reasoning, 2025. URL https://arxiv.org/abs/2510.09133.

Zhang, A., Chen, Y., Pan, J., Zhao, C., Panda, A., Li, J., and He, H. Reasoning models know when they're right: Probing hidden states for self-verification, 2025. URL https://arxiv.org/abs/2504.05419.

# A. Extended experiment specifications.

## A.1. Signal Extraction

We evaluate confidence signals that measure uncertainty or confidence at each thought chunk. Specifically, we focus on two primary metrics: Entropy After `</think>` (Wang et al., 2025b, EAT) and Confidence (Yang et al., 2025).

Let $p_\theta(\cdot \mid x)$ denote the next-token distribution at prefix $x$. We define the transformed prefix $x_{\text{forced}}$ by appending both, the thought termination tag and a forcing string.

$$x_{\text{forced}} = x \oplus \texttt{</think>} \oplus \texttt{forcing\_string} \tag{18}$$

where `forcing_string`, as described in the main text, is defined as `"\n **Final Answer**\n \boxed{"`

**Confidence** greedily rollouts an answer after $x_{\text{forced}}$, denoted as $\mathbf{a} = (a_1, \ldots, a_L)$. Then the length-normalized log likelihood over $\mathbf{a}$ is used as the confidence score

$$C_{\text{seq}}(x, \mathbf{a}) = \frac{1}{L} \sum_{i=1}^{L} \log p_\theta(a_i \mid x, a_{<i}). \tag{19}$$

**EAT** does not generate rollout, instead it directly looks at the entropy of the next token prediction distribution after $x_{\text{forced}}$:

$$\mathbb{H}(p_\theta(\cdot \mid x_{\text{forced}})). \tag{20}$$

**Probe signals.** We additionally compute probe-based signals trained on AIME reasoning trajectories.

Representation extraction and labeling: Let $P$ denote the original problem prompt and $x_{1:T}$ the trajectory. Suppose the model emits $K$ candidate final answers at token indices $1 \le t_1 < \cdots < t_K \le t_T$, with the $s$-th candidate answer denoted $\hat{a}_s$. For each step $s$, we reconstruct the decoding context

$$C_s = [P;\, x_{1:t_s}],$$

and extract the hidden representation of the last token at the last hidden layer:

$$h_s \in \mathbb{R}^d.$$

Each step is labeled according to correctness relative to the gold answer $a^\star$:

$$y_s = I[\hat{a}_s = a^\star] \in \{0, 1\}.$$

This yields a dataset

$$\mathcal{D} = \{(h_s, y_s)\}_{s=1}^{K}.$$

Probe model and training: We train a two-layer MLP probe (Zhang et al., 2025) to predict stepwise correctness $y_s$ from the representation $h_s$. The probe is trained on AIME 1983–2024.

# B. Risk control and finite-sample correction

This section details how we calibrate threshold parameters using distribution-free risk control. The goal is to select a signal–threshold pair (or parameter) that (i) satisfies a user-specified risk budget $\epsilon$, and (ii) among all feasible candidates, minimizes the corresponding efficiency loss.

## B.1. Problem setup

Let $\mathcal{V} = \{(x_i, y_i^*)\}_{i=1}^n$ be a labeled validation set. For a given uncertainty signal $s \in \mathcal{S}$ and a threshold parameter $\lambda \in \Lambda_s$ (e.g., $\lambda = \lambda_+$ for the upper threshold, or $\lambda = c$ for the parametric lower threshold), let $\tau_i(s, \lambda)$ denote the induced exit time on instance $i$. Given an instance-wise loss $\ell(\cdot; \lambda) \in [0, 1]$ (e.g., Eq. (6) or Eq. (7)), the population risk is

$$\text{Risk}(s, \lambda) \;=\; \mathbb{E}\big[\ell\big(y^*, f_{\tau(s,\lambda)}(x), \tilde{s}_{\tau(s,\lambda)}(x); \lambda\big)\big]\,,$$

and its empirical estimate on $\mathcal{V}$ is

$$\widehat{\text{Risk}}(\mathcal{V}, s, \lambda) \;=\; \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i^*, f_{\tau_i(s,\lambda)}(x_i), \tilde{s}_{\tau_i(s,\lambda)}(x_i); \lambda\big)\,. \tag{21}$$

### B.2. Why finite-sample correction is needed

Using $\widehat{\mathrm{Risk}}(\mathcal{V}, s, \lambda)$ directly can be over-optimistic due to sampling noise: a candidate $(s, \lambda)$ that appears feasible on $\mathcal{V}$ may violate the target risk after deployment. This issue is amplified when scanning a large grid $\bigcup_{s \in \mathcal{S}} \Lambda_s$ and selecting the most efficient feasible configuration, since the selection procedure can exploit random downward fluctuations in the empirical risk.

To mitigate this, we replace the raw empirical estimate with a conservative, finite-sample-adjusted quantity $\widetilde{\mathrm{Risk}}(\mathcal{V}, s, \lambda)$, and enforce feasibility using $\widetilde{\mathrm{Risk}}(\mathcal{V}, s, \lambda) \leq \epsilon$.

### B.3. Calibration methods

We consider two variants that differ only in how the risk constraint is enforced.

**Naive (no finite-sample correction).**

$$\widetilde{\mathrm{Risk}}_{\mathrm{naive}}(\mathcal{V}, s, \lambda) \;=\; \widehat{\mathrm{Risk}}(\mathcal{V}, s, \lambda). \tag{22}$$

This baseline is often effective when $n$ is large, but provides no protection against validation overfitting.

**UCB: concentration-based upper confidence bound.** For losses bounded in $[0, 1]$, Hoeffding's inequality implies that with probability at least $1 - \delta$,

$$\mathrm{Risk}(s, \lambda) \;\leq\; \widehat{\mathrm{Risk}}(\mathcal{V}, s, \lambda) \;+\; \sqrt{\frac{\log(1/\delta)}{2n}},$$

for a fixed $(s, \lambda)$. We therefore define

$$\widetilde{\mathrm{Risk}}_{\mathrm{UCB}}(\mathcal{V}, s, \lambda) \;=\; \widehat{\mathrm{Risk}}(\mathcal{V}, s, \lambda) \;+\; \sqrt{\frac{\log(1/\delta)}{2n}}, \tag{23}$$

and declare $(s, \lambda)$ feasible only if $\widetilde{\mathrm{Risk}}_{\mathrm{UCB}}(\mathcal{V}, s, \lambda) \leq \epsilon$. Operationally, the correction is larger when $n$ is small and vanishes as $n$ grows, yielding an adaptive conservativeness.

**Practical remark (multiple comparisons).** Eq. (23) provides a high-probability bound for a *fixed* candidate. When scanning a finite grid $\mathcal{G} = \{(s, \lambda) : s \in \mathcal{S}, \lambda \in \Lambda_s\}$, one may strengthen the bound via a union bound by replacing $\delta$ with $\delta / |\mathcal{G}|$. In our experiments, we use the simple correction above and report realized risks on a held-out test set.

## C. Ablation study results

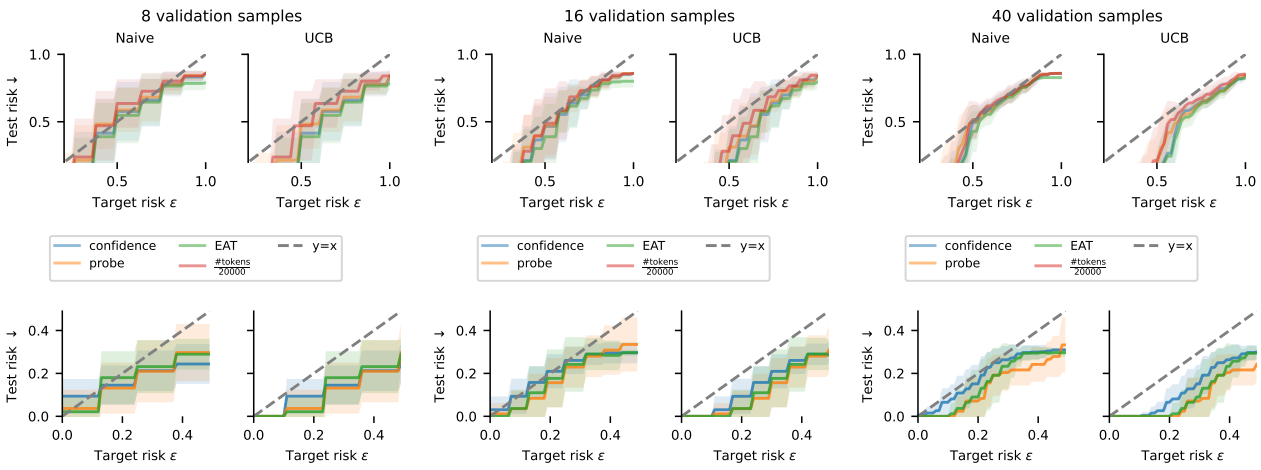



*Figure 7.* Ablation on validation set size. **Top row:** False positive risk; **Bottom row:** False negative risk. Principled risk control approach (UCB) shows better risk control than Naive cross-validation under small validation set size, as well as for

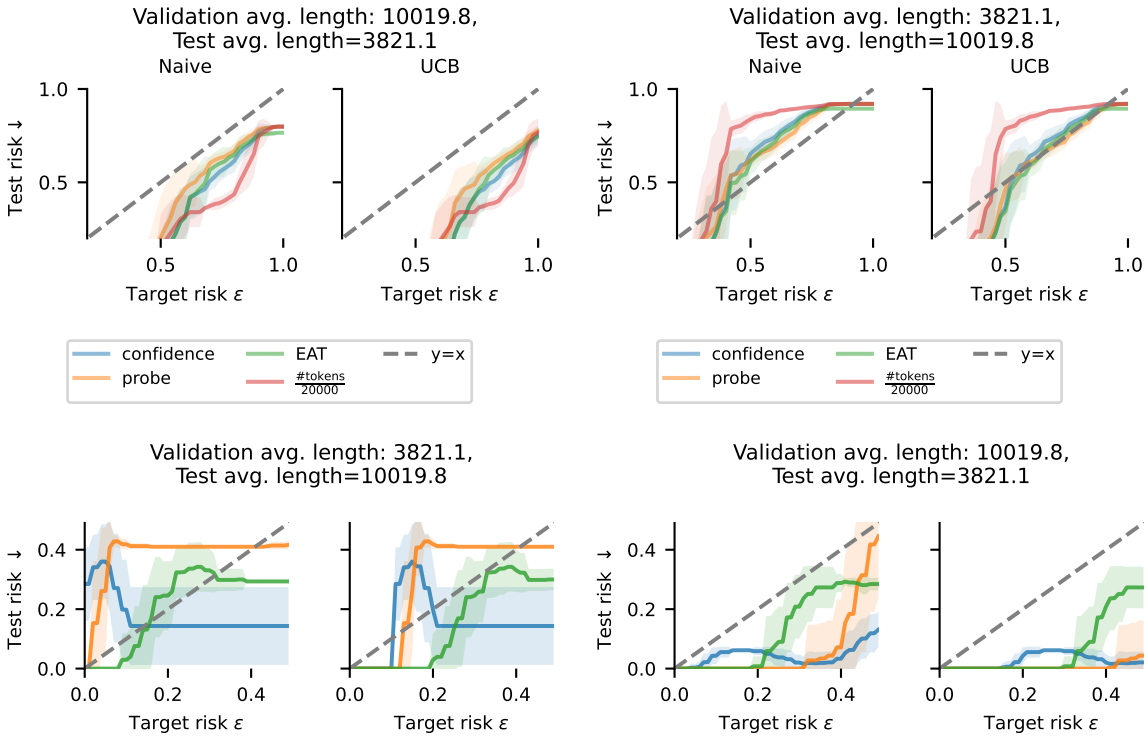



*Figure 8.* Ablation on length shift between validation and test set. **Top row:** False positive risk; **Bottom row:** False negative risk. Short to long shift (first column) brings more challenges to risk control. For upper threshold (top row), principled risk control alleviates excessive risk for all signals except for token-based. False negative risk, controlled by the lower threshold, shows a lack of robustness against length shift, in that the shape of the lower threshold is dependent on the horizon of the reaosning.

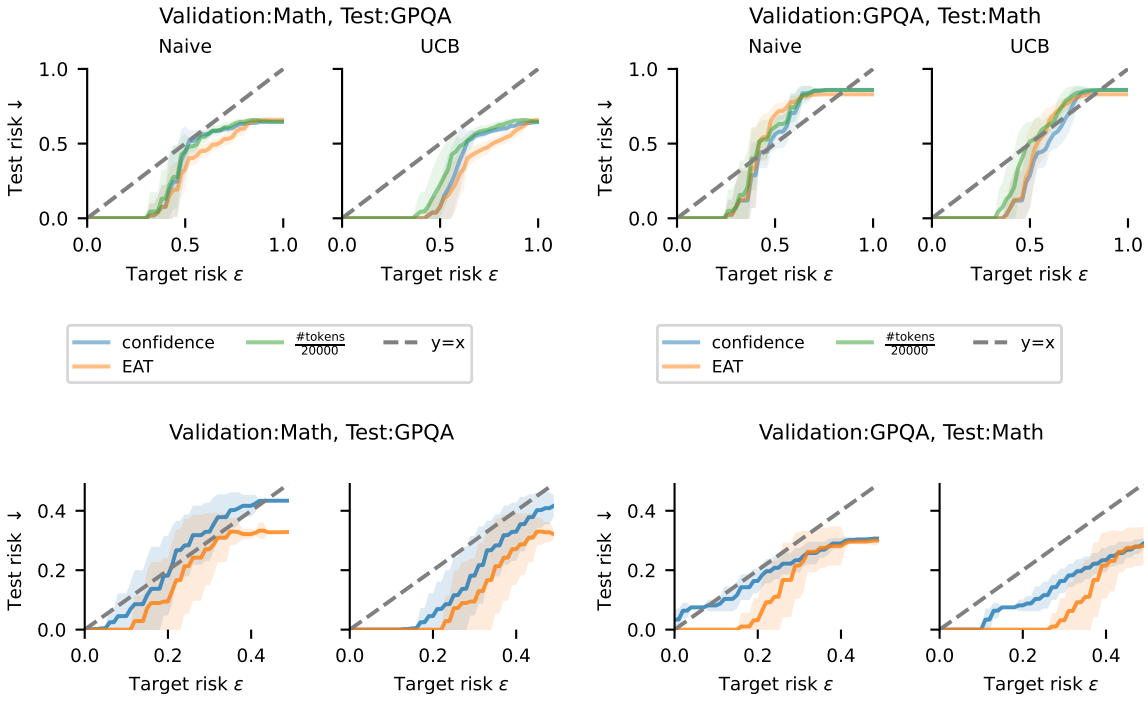



*Figure 9.* Ablation on dataset shift between validation and test set. **Top row:** False positive risk; **Bottom row:** False negative risk. Consistent with previous observations, using principled risk control again yields more controlled risk under distribution shift.