SAT: Balancing Reasoning Accuracy and Efficiency with Stepwise Adaptive Thinking

Weiyang Huang[1], Xuefeng Bai[1]*, Kehai Chen[1], Xinyang Chen[1], Yibin Chen[2], Weili Guan[1], Min Zhang[1]

[1] Institute of Computing and Intelligence, Harbin Institute of Technology, Shenzhen, China
[2] Huawei Technologies, China

Abstract

Large Reasoning Models (LRMs) have revolutionized complex problem-solving, yet they exhibit a pervasive "overthinking", generating unnecessarily long reasoning chains. While current solutions improve token efficiency, they often sacrifice fine-grained control or risk disrupting the logical integrity of the reasoning process. To address this, we introduce **S**tepwise **A**daptive **T**hinking (**SAT**), a framework that performs step-level, difficulty-aware pruning while preserving the core reasoning structure. SAT formulates reasoning as a Finite-State Machine (FSM) with distinct thinking modes (SLOW, NORMAL, FAST, SKIP). It navigates these states dynamically using a lightweight Process Reward Model (PRM), compressing easy steps while preserving depth for hard ones. Experiments across 9 LRMs and 7 benchmarks show that SAT achieves up to 40% reduction in reasoning tokens while generally maintaining or improving accuracy. Code is available at https://github.com/byxw13/SAT_Code.

1 Introduction

The advent of large reasoning models (LRMs) such as OpenAI-O1 (OpenAI, 2025) and DeepSeek-R1 (DeepSeek-AI et al., 2025) has marked a significant advance in tackling complex reasoning tasks (Wang et al., 2025b). Nevertheless, these models are prone to a notable "overthinking" issue: they tend to produce excessively long Chains of Thought (COT) even for straightforward questions, creating a practical bottleneck in real-time or resource-constrained applications (Sui et al., 2025).

To mitigate this problem, numerous studies have sought to steer LLMs toward more token-efficient reasoning. Earlier work employs strategic prompting (Renze and Guven, 2024; Chen et al., 2024; Xu et al., 2025), question-driven routing (Liang et al., 2025; Liu et al., 2025b), planning and budgeting (Han et al., 2025; Lin et al., 2025a). However, these methods often regulate the reasoning process at a coarse-grained level, offering limited precise control over both the reasoning flow and its efficiency. A more recent and promising direction explores dynamic decoding mechanisms, such as early stopping of the COT sequence (Liu and Wang, 2025; Yang et al., 2025b), or suppressing reflective steps (Huang et al., 2026; Wang et al., 2025a). While effective in shortening COT, these methods rely on a uniform policy across all steps, ignoring their varying difficulty and necessity. As shown in Figure 1, this strategy risks disrupting or cutting off essential steps, thereby compromising logical integrity and overall performance.

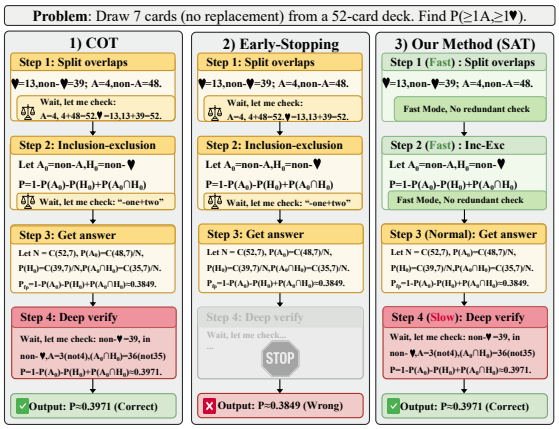

Figure 1: **Baselines vs. SAT.** COT spends redundant checks for easy steps, Early-Stopping halts after a high-confidence first-pass answer and fails, while SAT skips redundancy on easy steps but preserves verification on hard steps, achieving the correct answer efficiently.

In this work, we propose SAT (Stepwise Adaptive Thinking), a step-level, difficulty-aware pruning that eliminates redundancy while rigorously preserving the core logical structure essential for correct answers. As shown in Figure 1, departing from one-size-fits-all strategies, SAT is premised

*Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 43384–43402

on the observation that reasoning steps within a solution vary in difficulty (Saha et al., 2025).[1] Consequently, it dynamically navigates LLMs to employ a deeper thinking mode for challenging steps and a shallower, more efficient mode for simpler ones. Specifically, as illustrated in Figure 2, SAT formulates the reasoning process as a Finite-State Machine (FSM) wherein each reasoning step is intrinsically linked to an intermediate thinking state categorized into one of four distinct thinking modes: SLOW, NORMAL, FAST, and SKIP. These modes represent varying degrees of reasoning depth and computational effort. During inference, SAT dynamically navigates through this state space based on the evolving historical reasoning context, allowing for the selective compression or deliberate elaboration of the COT sequence.

To estimate the appropriate thinking state at each step, we introduce a step-level difficulty score based on progress reward and then develop a lightweight Process Reward Model (PRM, Lightman et al., 2024) for estimating the difficulty score based on confidence pattern and semantic information from the historical reasoning context. Leveraging these difficulty scores, SAT dynamically steers the reasoning trajectory by transitioning between states: favoring concise modes such as FAST when the reasoning step is deemed "easy", and invoking more exhaustive modes like NORMAL and SLOW for "complex" steps, while resorting to SKIP in overly difficult cases to guide the LLM toward an early, succinct conclusion. This control is achieved by injecting targeted prompting signals into the LLM input, directing it toward the desired reasoning depth. The framework offers two key advantages: (1) it gains efficiency from pruning redundancy without compromising the reasoning depth essential for correctness; (2) it functions as a lightweight, real-time "navigator" for the LLM's reasoning process, requiring no modifications to the base LLM's parameters and introducing minimal computational overhead.

We systematically evaluate the proposed method across 9 large reasoning models and 7 widely-used established benchmarks, covering mathematical reasoning, scientific reasoning, and programming tasks. Experimental results show that SAT achieves better efficiency-performance balance compared to strong baselines. Further analysis demonstrates that SAT incurs negligible overhead to achieve a 37% end-to-end speedup, while exhibiting difficulty-aware adaptivity that dynamically aligns reasoning depth with step difficulty, generalizing effectively across diverse domains.

Our contributions are summarized as follows:

- We propose SAT, a state machine-based framework that prunes redundancy at the step-level without compromising reasoning completeness.
- We design a lightweight PRM that reduces parameters by 99%, rendering real-time reasoning navigation practically feasible.
- Experiments on seven benchmarks demonstrate that our method achieves up to 40% token compression while generally maintaining accuracy.

2 Related Work

Mitigating Overthinking. Existing strategies for mitigating the "overthinking" problem at inference time can be broadly categorized into two groups, based on when and how they intervene in the reasoning process. **Static Strategies** impose constraints or select paths before generation. In terms of strategic prompting, Concise COT (Renze and Guven, 2024) explicitly instructs the model to be concise using few-shot demonstrations; Chen et al. (2024) feed the model with pre-computed solution templates matched to the estimated task difficulty. Distinctly, question-driven routing (Liang et al., 2025; Liu et al., 2025b) classify queries into discrete difficulty levels (e.g., simple, hard) at the onset to assign corresponding reasoning strategies. Planning and Budgeting (Han et al., 2025) estimates a token budget beforehand and prompts the model to solve the problem within this limit. However, these approaches rely on static, coarse-grained estimates that offer limited precise control over both the reasoning flow and its efficiency. **Dynamic Interventions** intervene directly during inference. Early stopping methods (Liu and Wang, 2025; Yang et al., 2025b; Lin et al., 2025b) seek to halt generation upon identifying moments of "sufficient certainty", while suppression-based methods (Wang et al., 2025a; Huang et al., 2026) mitigate redundancy by lowering the sampling probability of reflection-triggering tokens. While adaptive, these methods tend to *disrupt* the intrinsic reasoning process (e.g., truncating essential self-corrections or logical chains), degrading reasoning performance.

Instead of applying uniform compression or

[1] See Appendix B for comparisons with suppressing reflective methods.

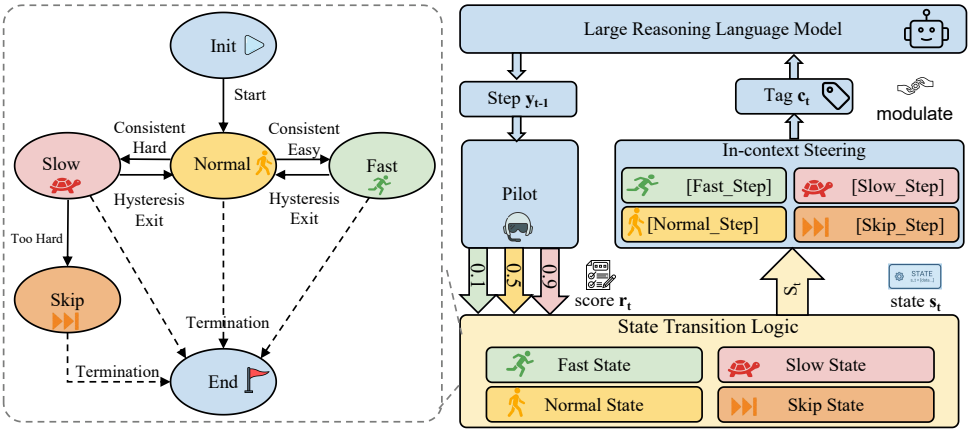

Figure 2: **Overview of the SAT framework.** The left panel provides a detailed view of the specific state transition dynamics and stability rules of the proposed reasoning FSM. The right panel illustrates the closed-loop control flow: the **Pilot** perceives the difficulty (r_t) of the previous step (y_{t-1}), driving the **State Transition Logic** to update the thinking state (s_t). This state is then mapped to a control tag (c_t) via **In-Context Steering** to modulate the next generation step (y_t).

global early stopping, this work dynamically assesses the difficulty of each reasoning step and selectively prunes only redundant segments, thereby preserving the essential logical progression of the COT while achieving significant efficiency gains.

Process Reward Models. Process Reward Models (PRMs; Lightman et al. 2024) have emerged as a pivotal technique for enhancing the reliability of reasoning in large models. By assigning quality scores to intermediate reasoning steps, they facilitate process-supervised training of LRMs (Luo et al., 2025). Beyond training, PRMs have also been effectively combined with inference-time search strategies—such as Tree of Thoughts (Wu et al., 2025), Best-of-N (Sun et al., 2024), and beam search (Wang et al., 2025f)—to prune erroneous reasoning paths and identify promising trajectories. However, PRM deployment is constrained by high computational cost, as existing methods typically rely on external, heavyweight verifiers (e.g., 7B parameters) which require costly computations (Ding et al., 2024). This makes them prohibitively expensive for efficient test-time scaling.

In contrast, we leverage confidence patterns as the primary discriminative feature to construct a lightweight PRM (30M parameters) that reduces parameters by 99% versus standard verifiers. This compact model is seamlessly integrated into our reasoning framework, where it monitors a single trajectory in real time and adaptively modulates reasoning density with negligible latency. Additional discussion of post-training approaches for mitigating overthinking is provided in Appendix A.

3 Methodology

Problem Formulation. Given an input query q, a LRM produces a COT trajectory with T steps, $Y = \{y_1, y_2, \ldots, y_T\}$, and finally outputs an answer a. The t-th step is generated solely conditioned on the preceding context:

$$y_t \sim P(\cdot \mid q, y_{<t}). \tag{1}$$

Following Lightman et al. (2024), each reasoning step y_t is defined as a text segment delimited by a newline character `\n` in the generated COT. This static paradigm allocates a *uniform thinking mode* to all steps, regardless of their difficulty, which often leads to "overthinking" for simpler steps.

3.1 Overview of the Reasoning FSM

In this work, we frame the reasoning process as a Finite-State Machine (FSM) (Figure 2). Unlike static approaches, this machine dynamically consumes the reasoning history and produces the subsequent reasoning steps. The reasoning FSM is defined as a hexad $\mathcal{M} = \langle \mathcal{S}, \Sigma, \Gamma, s_0, \delta, \omega \rangle$:

- $\mathcal{S}$ is the set of **thinking states** (including the initial state s_0=`INIT`) regulating reasoning depth.
- Σ is the **input space** of text segments; the input is the previous step $y_{t-1} \in \Sigma$.
- Γ is the **output space** ($\Gamma \equiv \Sigma$), where the output is the current step y_t.
- $\delta : \mathcal{S} \times \Sigma \to \mathcal{S}$ is the **transition function**, driven by the Pilot to update the state based on the semantic difficulty of y_{t-1}.

- $\omega : \mathcal{S} \to \Gamma$ is the **emission function**, where the LRM generates the next step y_t conditioned on the current state s_t.

State Space $\mathcal{S}$. We define the state space as $\mathcal{S} = \{\texttt{INIT}, \texttt{NORMAL}, \texttt{FAST}, \texttt{SLOW}, \texttt{SKIP}, \texttt{END}\}$. Beyond the boundary states `INIT` and `END`, we design four modes to modulate the **emission function** ω: `NORMAL` maintains standard depth; `FAST` accelerates simple steps by omitting redundancy; `SLOW` handles complex steps via detailed thinking; and `SKIP` acts as a "soft" termination, guiding the model to conclude naturally rather than rigid truncation.

State Transition δ. At each step t, the transition function updates the machine's state by analyzing the step difficulty of the incoming text segment y_{t-1}. Functionally, we decompose δ into a perception phase and a decision phase. First, the Pilot serves as the perception kernel, mapping the textual input y_{t-1} to a latent difficulty score $r_t \in [0, 1]$. Second, the state is updated based on this score and a history window $\mathcal{H}_k = \{r_{t-k+1}, \ldots, r_{t-1}\}$:

$$r_t = 1 - \text{Pilot}(y_{t-1}), \; s_t = f(s_{t-1}, r_t, \mathcal{H}_k). \quad (2)$$

Section 3.2 details the Pilot architecture, and Algorithm 1 outlines the specific update rules.

Emission Function ω. This function governs the generation of the reasoning step y_t by steering the LRM with a state-specific control token c_t:

$$c_t = \text{Tag}(s_t), \quad y_t \sim P(\cdot \mid q, y_{<t}, c_t). \quad (3)$$

The specific implementation of the tag mapping and steering prompts is detailed in Section 3.3.

3.2 Pilot: Lightweight Stepwise Difficulty Estimation

Stepwise Difficulty Estimation. To instantiate the perception phase of the transition function δ, we derive a difficulty score $r_t \in [0, 1]$ for the historical steps. Drawing on prior research that assesses problem-level difficulty based on the pass rate observed across multiple sampling trials (Tong et al., 2024), we extend this principle to the step level. Specifically, we assess the difficulty by the likelihood that the current step leads to a correct final solution. This objective aligns perfectly with the core function of Process Reward Models (PRMs) (Wang et al., 2024b), which are explicitly designed to estimate this probability of correctness, denoted as v_t. Consequently, we define the difficulty of the current step as the complement of its success probability:

$$r_t = 1 - v_t. \quad (4)$$

Intuitively, a lower probability of success (a lower PRM score) indicates a higher difficulty level, thereby necessitating deeper reasoning resources.

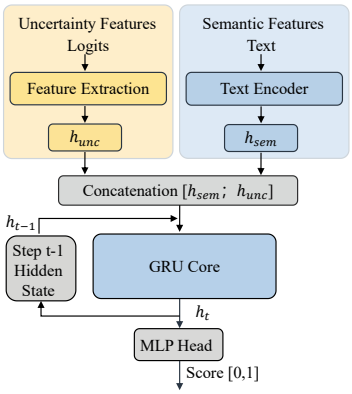

Figure 3: **Lightweight Pilot.** A GRU-based framework fusing semantic embeddings and uncertainty features to capture stepwise difficulty trajectories.

Lightweight Pilot. Most existing PRMs are built on billion-parameter generative backbones (e.g., Qwen-2.5-7B), making them hard to be adapted to efficient reasoning. To address this, we introduce a 30M-parameter Pilot that removes generative overhead while retaining accurate difficulty estimation, yielding ∼99% lower parameters than typical generative PRMs and retaining over 80% of the performance of standard LLM-based PRMs.

Architecture. As illustrated in Figure 3, the Pilot maps the input y_{t-1} into a fused representation $\mathbf{z}_t$ via two channels: **Uncertainty Features ($\mathbf{h}_{\text{unc}}$)** derived from generation logits (e.g., entropy; see Appendix C) to quantify internal confidence, and **Semantic Features ($\mathbf{h}_{\text{sem}}$)** obtained by encoding the text content via an encoder (GTE-small, Li et al., 2023). These features are concatenated $\mathbf{z}_t = [\mathbf{h}_{\text{unc}}; \mathbf{h}_{\text{sem}}]$ and processed by a GRU, projecting the hidden state to a scalar correctness score v_t via learnable parameters $\mathbf{w}$ and b:

$$\mathbf{h}_t = \text{GRU}(\mathbf{z}_t, \mathbf{h}_{t-1}), \quad v_t = \sigma(\mathbf{w}^\top \mathbf{h}_t + b). \quad (5)$$

The v_t is converted to the difficulty score r_t, completing the perception phase of the state transition.

Teacher-Student Distillation. We train the Pilot by distilling a teacher PRM. We generate step trajectories on PRM800K (Lightman et al., 2024) using DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI et al., 2025) and obtain teacher probabilities v_t^* from Skywork-o1-Open-PRM-7B (He et al., 2024).

Algorithm 1 FSM Inference Step

Function: Taking previous step y_{t-1}, state s_{t-1}, and history $\mathcal{H}$ as inputs, this algorithm executes one reasoning cycle: it perceives the difficulty of y_{t-1}, updates state s_t, and **generates the next reasoning step** y_t via the function ω.

if y_{t-1} contains `</think>` **then return** $(\texttt{END}, \varnothing)$
// 1. Perception Phase (Pilot)
$r_t \leftarrow 1 - \text{Pilot}(y_{t-1})$ ▷ Extract latent difficulty from text
Update history window $\mathcal{H}$ with r_t
// 2. Transition Logic (Dynamics)
$s_t \leftarrow s_{t-1}$
if $s_{t-1} = \texttt{INIT}$ **then** $s_t \leftarrow \texttt{NORMAL}$
if $s_{t-1} = \texttt{NORMAL}$ **then** ▷ Consistent Entry
    if $\text{All}(\mathcal{H} < \tau_{\texttt{fast}})$ **then** $s_t \leftarrow \texttt{FAST}$
    if $\text{All}(\mathcal{H} > \tau_{\texttt{slow}})$ **then** $s_t \leftarrow \texttt{SLOW}$
if $s_{t-1} \in \{\texttt{FAST}, \texttt{SLOW}\}$ **then** ▷ Hysteresis Exit
    if Score crosses $\tau \pm \Delta$ **then** $s_t \leftarrow \texttt{NORMAL}$
if $s_t = \texttt{SLOW} \wedge \text{All}(\mathcal{H} > \tau_{\texttt{skip}})$ **then** $s_t \leftarrow \texttt{SKIP}$
// 3. Emission Phase (Generation)
$c_t \leftarrow \text{Tag}(s_t)$ ▷ Determine control token
$y_t \leftarrow \text{Generate}(q, y_{<t}, c_t)$ ▷ Generate next step via ω
return (s_t, y_t)

The Pilot is optimized with Binary Cross-Entropy to align its predicted correctness v_t:

$$\mathcal{L} = -\mathbb{E}\left[v_t^* \log v_t + (1 - v_t^*) \log(1 - v_t)\right]. \quad (6)$$

The distilled Pilot achieves high correlation with the teacher (as shown in Section 4.3) at orders-of-magnitude lower cost, enabling real-time feedback for online FSM control. Crucially, since we freeze the gte encoder (∼30M parameters) and only update the GRU (∼0.15M parameters), the entire training process **converges in less than 5 minutes on a single GPU**. This renders the deployment of SAT highly accessible and orders of magnitude more efficient than training generative verifiers.

3.3 Stepwise Difficulty Guided FSM

Building on the stepwise difficulty score, we instantiate the FSM's **transition function** δ with a difficulty-aware algorithm, and realize the **emission function** ω via in-context steering.

Difficulty-aware Transition. The transition logic is specified in Algorithm 1. At each step boundary, the FSM first checks the termination condition. If the process continues, the state s_t is updated based on the difficulty history $\mathcal{H}_k$ and a set of pre-defined **thresholds** $\mathcal{T} = \{\tau_{\texttt{fast}}, \tau_{\texttt{slow}}, \tau_{\texttt{skip}}\}$. The logic follows two key principles: (i) *Consistent Entry*: The system transitions from `NORMAL` to `FAST` or `SLOW` only when the difficulty scores consistently breach the corresponding **threshold** (i.e., $\text{All}(\mathcal{H}_k < \tau_{\texttt{fast}})$). (ii) *Hysteresis Exit*: To prevent state flickering, returning to `NORMAL` requires the score r_t to cross the **threshold** with a safety margin Δ. Additionally, if the difficulty remains excessively high while in the `SLOW` state, the state shifts to `SKIP` to encourage consolidation.

Emission via In-Context Steering. To implement the emission function ω, we steer the generation by manipulating the input context of the LRM. Specifically, we append the state-specific control tag $c_t = \text{Tag}(s_t)$ to the current reasoning context:

$$\text{Context}_{t+1} = \text{Context}_t \oplus \text{Token}(c_t). \quad (7)$$

This explicitly modulates the model's decoding distribution to align with the semantic requirements of s_t (e.g., [Fast_Step] triggers concise decoding), without requiring architectural modifications. A full list of tags are provided in Appendix D.

4 Experiment

4.1 Experimental Setup

Datasets and Evaluation Metrics. To comprehensively evaluate SAT's performance across diverse domains and difficulty levels, we conduct experiments on 7 benchmarks covering mathematical, scientific, and code reasoning. For **mathematical reasoning**, we evaluate general capabilities on GSM8K (Cobbe et al., 2021) and MATH 500 (Hendrycks et al., 2021), and competition-level performance on AMC 2023 (AI-MO, 2024), AIME 2024 (MAA Committees, 2024) and AIME 2025 (MAA Committees, 2025). For SAT on AMC, AIME 2024, and AIME 2025, we report the average over three runs due to the small benchmark size. For **scientific reasoning**, we utilize the expert-level GPQA Diamond (Rein et al., 2024). For **code reasoning**, we adopt HumanEval (Chen et al., 2021). We report Accuracy (**Acc**, Pass@1) and average generated tokens per query (**Token**) as evaluation metrics.

Backbone LRMs and Baselines. We evaluate SAT across diverse backbones: Qwen3 series (Yang et al., 2025a), DeepSeek-R1-Distill-Qwen series (DeepSeek-AI et al., 2025), Llama-3.1-Nemotron-8B (Bercovich et al., 2025) and QwQ-32B (QwenTeam, 2025). We compare SAT against three categories of baseline methods: (1) **COT**: Chain-of-thought reasoning; (2) **DEER** (Yang et al., 2025b): the SoTA early-exit method based on confidence truncation; (3) **CGRS** (Huang et al., 2026): the SoTA suppression method that suppresses the sampling of reflection-triggering tokens. (4) **ThinkSwitcher** (Liang et al., 2025): a recent *question-driven routing* approach that uses a switcher to select between short and long

Model	Method	GSM8K		MATH500		AIME 2024		AIME 2025		AMC	
		Acc (↑, %)	Tokens (↓)	Acc (↑, %)	Tokens (↓)	Acc (↑, %)	Tokens (↓)	Acc (↑, %)	Tokens (↓)	Acc (↑, %)	Tokens (↓)
Qwen3-4B	COT	**95.1**	2136	95.0	4823	60.0	12662	50.0	12720	92.5	7408
	DEER	94.5 (-0.6)	1250 (-41%)	92.6 (-2.4)	3214 (-33%)	**63.3 (+3.3)**	9327 (-26%)	55.0 (+5.0)	12084 (-5%)	87.5 (-5.0)	4906 (-34%)
	CGRS	–	–	91.3 (-3.7)	**2704 (-44%)**	56.7 (-3.3)	**7893 (-37%)**	–	–	86.7 (-5.8)	**4351 (-41%)**
	SAT	**95.1 (+0.0)**	**845 (-60%)**	**95.6 (+0.6)**	2833 (-41%)	61.1 (+1.1)	9184 (-27%)	**56.6 (+6.6)**	**10228 (-20%)**	**93.3 (+0.8)**	5145 (-31%)
Qwen3-8B	COT	**95.8**	2152	95.6	5166	66.7	12393	**60.0**	12835	90.0	7920
	DEER	95.5 (-0.3)	981 (-54%)	92.6 (-3.0)	**2732 (-47%)**	61.7 (-5.0)	8796 (-29%)	**60.0 (+0.0)**	12229 (-5%)	**92.5 (+2.5)**	**4392 (-45%)**
	CGRS	–	–	93.3 (-2.3)	3507 (-32%)	61.1 (-5.6)	**8792 (-29%)**	–	–	89.2 (-0.8)	5595 (-29%)
	SAT	95.5 (-0.3)	**879 (-60%)**	**95.8 (+0.2)**	3215 (-38%)	**68.9 (+2.2)**	9281 (-25%)	58.9 (-1.1)	**10909 (-15%)**	90.0 (+0.0)	5062 (-36%)
Qwen3-14B	COT	95.7	1660	96.2	4598	73.3	11742	56.7	12820	92.5	6964
	DEER	95.3 (-0.4)	840 (-49%)	94.0 (-2.2)	3074 (-33%)	**76.7 (+3.4)**	**7619 (-35%)**	**66.7 (+10.0)**	11135 (-13%)	95.0 (+2.5)	**4763 (-32%)**
	CGRS	–	–	94.5 (-1.7)	3235 (-30%)	70.0 (-3.3)	8662 (-26%)	–	–	93.3 (+0.8)	5076 (-27%)
	SAT	**96.4 (+0.7)**	**767 (-54%)**	**96.6 (+0.4)**	**2904 (-37%)**	71.1 (-2.2)	9598 (-18%)	61.1 (+4.4)	**10630 (-17%)**	**95.8 (+3.3)**	4835 (-31%)
Qwen3-32B	COT	96.0	1688	95.6	4358	70.0	10788	63.3	12203	95.0	6448
	DEER	96.2 (+0.2)	769 (-54%)	94.2 (-1.4)	3418 (-22%)	**76.7 (+6.7)**	8682 (-20%)	66.7 (+3.4)	10893 (-11%)	**97.5 (+2.5)**	5753 (-11%)
	CGRS	–	–	93.1 (-2.5)	2993 (-31%)	65.6 (-4.4)	**8128 (-25%)**	–	–	94.2 (-0.8)	**4766 (-26%)**
	SAT	**96.5 (+0.5)**	**680 (-60%)**	**96.6 (+1.0)**	**2719 (-38%)**	70.0 (+0.0)	9306 (-14%)	**70.0 (+6.7)**	**10011 (-18%)**	95.8 (+0.8)	4911 (-24%)
DS-Qwen-1.5B	COT	77.6	1193	82.2	4723	**26.7**	11941	23.3	11879	67.5	7729
	DEER	74.7 (-2.9)	984 (-18%)	67.8 (-14.4)	**2497 (-47%)**	23.3 (-3.4)	9553 (-20%)	10.0 (-13.3)	9281 (-22%)	60.0 (-7.5)	**5496 (-29%)**
	ThinkSwitcher	**84.7** (+7.1)	2114 (+77%)	**82.4 (+0.2)**	4544 (-4%)	23.3 (-3.4)	**8192 (-31%)**	**28.3 (+5.0)**	**6689 (-43%)**	–	–
	SAT	77.3 (-0.3)	**564 (-53%)**	**82.4 (+0.2)**	3230 (-32%)	**26.7 (+0.0)**	10893 (-9%)	20.0 (-3.3)	9455 (-20%)	**68.3 (+0.8)**	6642 (-14%)
DS-Qwen-7B	COT	89.9	532	**92.8**	3537	50.0	12662	30.0	11028	87.5	6366
	DEER	90.6 (+0.7)	917 (+72%)	89.8 (-3.0)	2143 (-39%)	49.2 (-0.8)	9839 (-22%)	36.7 (+6.7)	7257 (-34%)	85.0 (-2.5)	4451 (-30%)
	CGRS	–	–	87.6 (-5.2)	**1867 (-47%)**	**52.2 (+2.2)**	**7597 (-40%)**	–	–	**88.3 (+0.8)**	**3406 (-46%)**
	ThinkSwitcher	**92.5 (+2.6)**	1389 (+161%)	91.3 (-1.5)	3495 (-1.0%)	48.3 (-1.7)	7936 (-37%)	**37.5 (+7.5)**	**6948 (-37%)**	–	–
	SAT	89.3 (-0.6)	**385 (-28%)**	**92.8 (+0.0)**	2237 (-37%)	51.1 (+1.1)	8336 (-34%)	36.7 (+6.7)	9496 (-14%)	**88.3 (+0.8)**	3642 (-43%)
DS-Qwen-14B	COT	94.9	1122	94.4	3539	60.0	10343	36.7	11002	92.5	5333
	DEER	93.3 (-1.6)	1040 (-7%)	89.8 (-4.6)	2577 (-27%)	**68.4 (+8.4)**	8115 (-22%)	36.7 (+0.0)	10125 (-8%)	85.0 (-7.5)	**4240 (-20%)**
	ThinkSwitcher	94.3 (-0.6)	1042 (-7%)	92.7 (-1.7)	3572 (+1%)	60.4 (+0.4)	**8044 (-22%)**	**42.5 (+5.8)**	10065 (-9%)	–	–
	SAT	**95.2 (+0.3)**	**621 (-45%)**	**94.8 (+0.4)**	**2515 (-29%)**	62.2 (+2.2)	8413 (-19%)	38.9 (+2.2)	**8912 (-19%)**	**93.3 (+0.8)**	4582 (-14%)
Nemo-Llama-8b	COT	**90.2**	1400	93.8	3412	56.7	10280	**40**	10893	90	5997
	DEER	89.8 (-0.4)	1473 (+5%)	91.4 (-2.4)	2995 (-12%)	**66.7 (+10)**	9755 (-5%)	36.7 (-3.3)	11820 (+9%)	90 (+0.0)	5408 (-10%)
	SAT	89.7 (-0.5)	**1035 (-26%)**	**94 (+0.2)**	**2844 (-17%)**	57.8 (+1.1)	**9065 (-12%)**	35.6 (-4.4)	**10555 (-3%)**	**95.8 (+5.8)**	**4610 (-23%)**
QwQ-32B	COT	**97.0**	1561	**97.0**	4025	66.7	11305	**63.3**	12554	90.0	7086
	DEER	96.3 (-0.7)	977 (-37%)	94.6 (-2.4)	3316 (-18%)	70.0 (+3.3)	10097 (-11%)	50.0 (-13.3)	11598 (-8%)	**95.0 (+5.0)**	5782 (-18%)
	CGRS	–	–	94.2 (-2.8)	**2810 (-30%)**	68.9 (-1.1)	**8202 (-27%)**	–	–	93.3 (+3.3)	**4771 (-33%)**
	SAT	96.6 (-0.4)	**969 (-38%)**	**97.0 (+0.0)**	3256 (-19%)	**71.1 (+4.4)**	9288 (-18%)	58.9 (-4.4)	**10849 (-14%)**	94.2 (+4.2)	6053 (-15%)

Table 1: **Comparison across diverse reasoning models.** Metrics include **Acc** (↑) and **Tokens** (↓). Changes relative to the COT are highlighted in orange for Acc and blue for Tokens. Best results within each group are bolded.

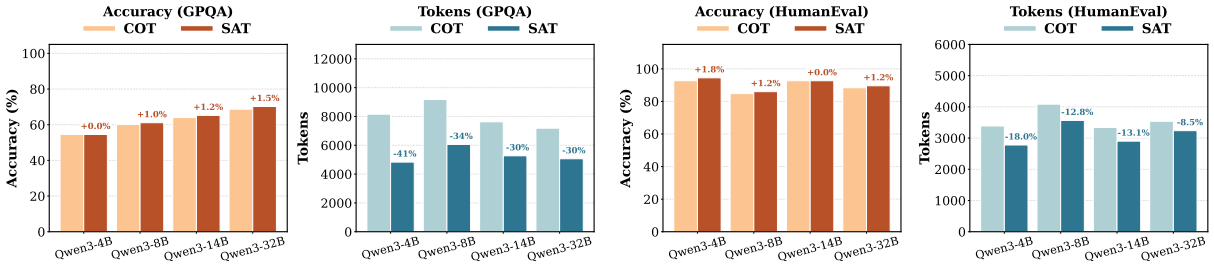

Figure 4: **Accuracy and token usage on GPQA and HumanEval across different model scales.**

COT. Due to space constraints, comparisons with additional methods are detailed in Appendix F.1.

Implementation Details. All our methods are implemented based on the *HuggingFace Transformers* framework and conducted on NVIDIA H20 (96GB) GPUs. We set sampling parameters to Temperature $= 0.6$, Top-$p = 0.95$ and a max length of 16,384 tokens. For the FSM, we set the difficulty thresholds to $\tau_{\texttt{fast}} = 0.2$, $\tau_{\texttt{slow}} = 0.6$. The hysteresis margin is set to $\Delta = 0.1$. Details are provided in Appendix E.

4.2 Main Results

Overall Performance. As summarized in Table 1, SAT demonstrates a generally superior accuracy–efficiency balance compared to Vanilla COT and other baselines across five mathematical reasoning benchmarks.[2] On average, over all tested models and datasets, SAT reduces token usage by 25.1% while improving accuracy by +1.5 points. In comparison, DEER truncates more aggressively—reducing tokens by 47% on MATH-500 (Qwen3-8B)—and can outperform SAT on some AIME settings, but its trade-off is less consistent overall, with accuracy drops in some cases (e.g., -3.0 on the same MATH-500 setting). Similarly, CGRS can introduce notable performance drops (e.g., –2.3 accuracy on MATH-500 for Qwen3-8B). ThinkSwitcher, as a question-driven

[2]See Appendix F.1 for more baseline comparisons.

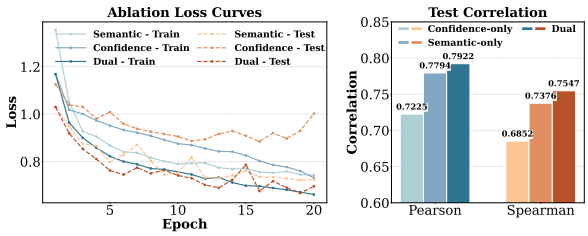

Figure 5: **Ablation study results.** Left: training and test loss curves for GRU-based Pilot under different inputs. Right: test-set correlation (Pearson/Spearman) between predicted step scores and supervision targets.

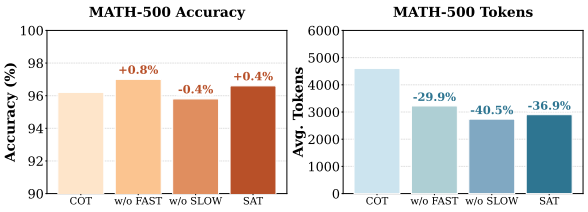

Figure 6: **Strategy ablation results on MATH500 (Qwen3-14B).** Accuracy and token usage of *COT*, *w/o FAST*, *w/o SLOW*, and SAT.

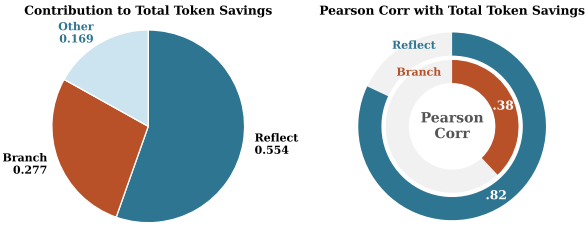

Figure 7: **Attribution analysis of token savings on MATH500 (Qwen3-14B).** Left: decomposition of SAT's token savings into reflection-related and branching-related components. Right: Correlation between per-sample token savings and each component.

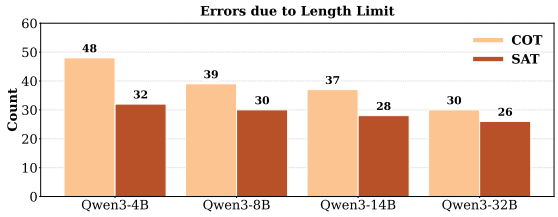

Figure 8: **Length-limit failures on aggregated math benchmarks.** Number of incorrect predictions caused by exceeding maximum length, aggregated over GSM8K, MATH500, AIME 2024/2025, and AMC.

routing baseline, can reduce computation on some datasets (e.g., **-43%** Tok on AIME 2025 for DS-Qwen-1.5B) but its gains are less consistent across tasks and models, underscoring the advantage of SAT 's stepwise online modulation.

Generalization across Domains and Model Scales. As illustrated in Figure 4, SAT maintains strong generalization across both scientific reasoning (GPQA Diamond) and code reasoning (HumanEval) tasks, achieving consistent efficiency improvements with nearly no compromise in accuracy. Moreover, SAT scales stably across models of varying sizes, achieving an average reduction of 33.8% in tokens on GPQA with a +0.75% gain in accuracy, and 13.1% fewer tokens on HumanEval alongside a +1.05% improvement in accuracy. We further extend evaluation to open-ended dialogue and smaller backbones: on MT-Bench, SAT maintains comparable response quality while reducing generation cost, and on Qwen3-0.6B, it preserves accuracy while consistently reducing tokens (Appendix F.2; Appendix F.3).

4.3 Ablation Study

Pilot Ablation: dual vs. single-source features. Figure 5 compares three variants of the Pilot module: *confidence-only* (relying solely on uncertainty features), *semantic-only* (using only GTE-small embeddings), and the *dual-input* model. Overall, the dual-input model consistently shows the best optimization behavior and test performance. While semantic features outperform confidence features alone, their combination yields better performance than either feature source alone. These results indicate that confidence features—capturing local uncertainty dynamics—and semantic features—encoding step content—provide complementary signals for difficulty estimation.

Strategy Ablation: FAST vs. SLOW. Figure 6 presents an ablation study on MATH500 (Qwen3-14B) to assess the contribution of each thinking mode. Removing the FAST mode (*w/o FAST*) yields a slight accuracy gain (+0.8% over COT), whereas disabling SLOW (*w/o SLOW*) leads to a small decline (-0.4%), indicating that SLOW reasoning is essential for reasoning. Meanwhile, SAT preserves accuracy while saving tokens (+0.4% on accuracy; -36.9% on token usage). All ablated variants reduce token usage, with *w/o FAST* saving 29.9% and *w/o SLOW* achieving the largest compression (40.5%, 2737). Interestingly, *w/o FAST* also reduces token cost; Appendix F.4 further shows it decreases reflective behaviors, and we hypothesize this comes from fewer "error–repair–re-verify" loops.

4.4 Analysis

Why can SAT save tokens? Analyzing Qwen3-14B on MATH500 (Figure 7), SAT's efficiency stems mainly from reduced reflection (contribution

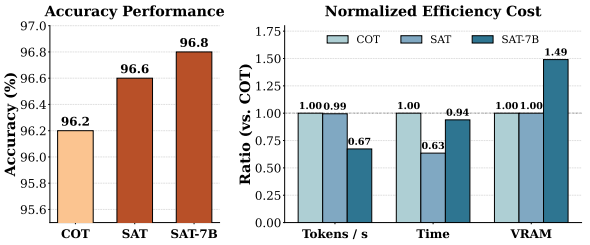

Figure 9: **Performance analysis on MATH500 (Qwen3-14B).** Left: Absolute accuracy comparison. Right: Normalized computational costs (Tokens/s, Time, VRAM) relative to the COT baseline (1.0×).

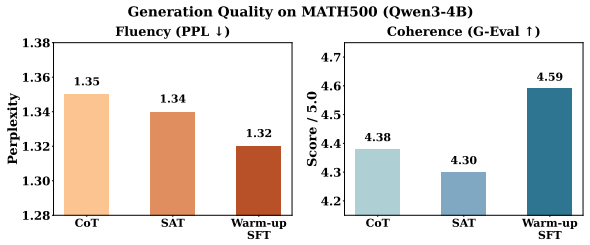

Figure 10: **Generation quality on MATH500 with Qwen3-4B.** Left: fluency (perplexity; lower is better). Right: coherence (G-Eval; higher is better).

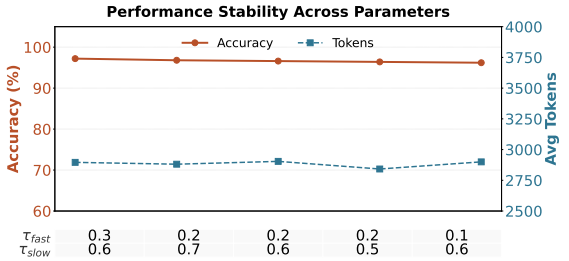

Figure 11: **Hyperparameter robustness on MATH500 (Qwen3-14B).** Sensitivity of SAT to $\tau_{\texttt{fast}}$ and $\tau_{\texttt{slow}}$.

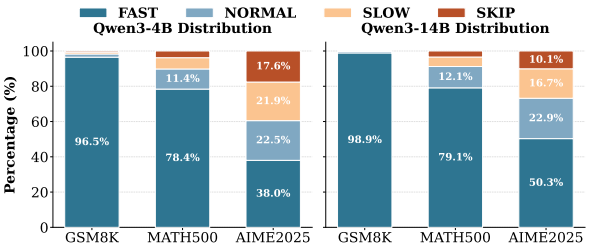

Figure 12: **State allocation across dataset difficulty and model scales.** We report the step-level thinking mode ratios on GSM8K (easy), MATH500 (medium), and AIME 2025 (hard), for Qwen3-14B and Qwen3-4B.

0.554, Pearson r=0.818) and secondarily from limited branching (contribution 0.277, Pearson r=0.374). This confirms that curbing redundant reflection is the primary driver of token savings, with branching control providing auxiliary benefits (See details settings in Appendix F.5).

Why can SAT slightly improve accuracy? Despite its efficiency focus, SAT achieves consistent accuracy gains by optimizing generation budgets. By pruning redundant reasoning, it prevents valid solutions from exceeding the context limit (16k). As shown in Figure 8, SAT reduces length-limit failures in the Qwen3 model series from **154** to **116** (**-24.7%**), effectively salvaging samples that would otherwise fail due to premature truncation. Moreover, by curbing error–repair–re-verify loops, SAT reduces stochastic drift from repeated self-corrections, yielding additional accuracy gains.

Does SAT Affect Generation Quality? We further evaluate generation quality on MATH500 using Qwen3-4B. As shown in Figure 10, SAT preserves fluency and keeps coherence comparable to COT, suggesting that stepwise token reduction does not noticeably harm output quality. We also test a lightweight Warm-up SFT strategy by fine-tuning Qwen3-4B on 2,000 LLM-annotated samples generated by applying SAT to Qwen3-32B, aiming to mitigate the train–test mismatch introduced by control tags. Warm-up SFT yields moderate quality gains, indicating potential benefit, but its cost is also clear: it requires backbone-specific fine-tuning and may introduce overfitting, with GPQA accuracy dropping from 54.5% to 44.9% in our primarity experiment. We therefore keep the main method focused on training-free deployment.

Additional Computation Cost. We assess the computational overhead of SAT by comparing it with standard COT and a variant using a 7B-scale pilot model (SAT-7B). As shown in Figure 9 (left), SAT retains COT's throughput (29.98 vs. 30.14 tokens/s) while matching the accuracy (∼96.6%) of the heavier baselines, validating the efficacy of the 30M PRM. Crucially, Figure 9 (right) reveals that while SAT-7B's overhead (1.5× VRAM) negates its savings, SAT maintains COT-level memory (∼30GB) and translates token reduction into a ∼37% end-to-end speedup. This confirms SAT achieves practical acceleration with negligible overhead (details in Appendix F.6).

Hyperparameter Robustness. We further analyze the choice of hyperparameter on MATH500 using Qwen3-14B. Figure 11 shows that SAT is highly robust to threshold choices: under mild perturbations of $(\tau_{\texttt{fast}}, \tau_{\texttt{slow}})$ (e.g., $\tau_{\texttt{fast}} \in \{0.3, 0.2, 0.1\}$ and $\tau_{\texttt{slow}} \in \{0.7, 0.6, 0.5\}$), accuracy remains consistently high (i.e., ≈96.6%±0.6), while average tokens stay in a narrow band (2842–2904;

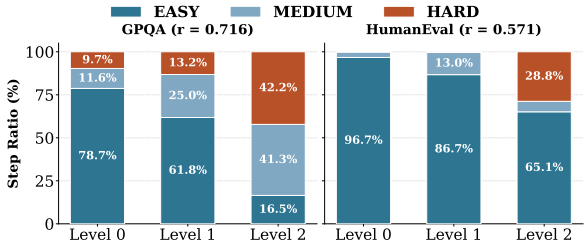

Figure 13: **Cross-domain generalization of the Pilot on GPQA and HumanEval.** Step-level scores are bucketed into Easy (< 0.2), Medium (0.2–0.6), and Hard (> 0.6) under a sampling-based difficulty proxy.

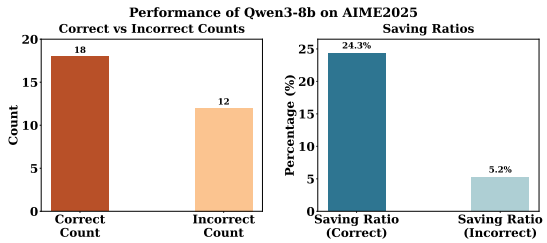

Figure 14: **Outcome-conditioned token savings on AIME2025 (Qwen3-8B).** SAT saves substantially more tokens on correct instances than on failed ones, where many runs hit the max-token limit.

≈2.1% relative range), indicating stable accuracy–efficiency trade-offs across settings.

Difficulty- and Scale-aware State Allocation. Figure 12 depicts how SAT dynamically shifts thinking modes in response to problem difficulty and model scale. While dominating GSM8K with FAST (**98.9%**), SAT progressively reallocates compute for challenging tasks: on AIME 2025, FAST drops to **50.3%** while SLOW+SKIP rises to **26.8%** (Qwen3-14B). Crucially, the behavior is also sensitive to model capability: the weaker Qwen3-4B triggers significantly more deep reasoning on AIME 2025 than 14B (**39.5%** vs. **26.8%**).

Cross-domain Generalization of the Pilot. We further test whether the Pilot generalizes beyond math on GPQA and HumanEval using a sampling-based difficulty proxy (details in Appendix F.7). Figure 13 shows a consistent monotonic trend in both domains: Easy steps decrease while Hard steps increase as proxy difficulty rises, with Pearson r=0.716 on GPQA and r=0.571 on HumanEval. Despite being trained only on mathematical data, the Pilot produces meaningful difficulty signals on scientific and code reasoning tasks.

Efficiency Gains are Outcome-Sensitive. Figure 14 reveals that savings are outcome-dependent: SAT saves **24.3%** on correct instances but only **5.2%** on incorrect ones (AIME 2025). This aligns with the intuition that hard problems demand extensive exploration, where intermediate detours are necessary rather than redundant. Notably, most failures (10/12) hit the max generation limit, indicating active search saturation that leaves minimal headroom for safe pruning on extreme difficulties.

5 Conclusion

We propose SAT, a framework that dynamically modulates reasoning depth via a lightweight, step-wise difficulty estimator. By navigating a Finite-State Machine, SAT decouples reasoning efficacy from computational cost, achieving substantial efficiency gains (up to 40% token reduction) without compromising accuracy. Our work validates that precise, step-level intervention is a viable and superior alternative to coarse-grained routing or static prompting for efficient reasoning deployment.

Limitations

We identify two limitations in our current framework. First, while the Pilot is designed to be lightweight, integrating an external module introduces a marginal computational overhead during the perception phase, although this is largely offset by the efficiency gains from token pruning. Second, our method leverages in-context steering to modulate reasoning depth. Consequently, the precise execution of the state-machine logic relies on the backbone model's inherent instruction-following capabilities.

Acknowledgments

This work was supported in part by the Science Fund for Creative Research Groups of the National Natural Science Foundation of China under Grant 62521006, in part by the National Natural Science Foundation of China (62406091, 62276077, 62306085, U23B2055, U24A20328, 62476071, 62350710797), in part by the Xinjiang Science and Technology Development Plan for the Two Innovation Demonstration Zones along the Silk Road Economic Belt under Grant 2024LQ03003, in part by Guangdong S&T Program (2024B0101050003), in part by the Guangdong Basic and Applied Basic Research Foundation (2026A1515011718, 2024A1515011205, 2025A1515011732), and in part by Shenzhen Science and Technology Program (KQTD20240729102154066).

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Second Conference on Language Modeling*.

AI-MO. 2024. AMC 2023, 2024. https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 114 others. 2025. Llama-nemotron: Efficient reasoning models. *Preprint*, arXiv:2505.00949.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Qiguang Chen, Libo Qin, Jiaqi WANG, Jingxuan Zhou, and Wanxiang Che. 2024. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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. 2025. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*.

Xiaoxue Cheng, Junyi Li, Zhenduo Zhang, Xinyu Tang, Xin Zhao, XinYu KONG, and Zhiqiang Zhang. 2025. Incentivizing dual process thinking for efficient large language model reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Everything of thoughts: Defying the law of penrose triangle for thought generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1638–1662, Bangkok, Thailand. Association for Computational Linguistics.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. 2025. Efficiently scaling LLM reasoning programs with certaindex. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Runquan Gui, Yafu Li, Xiaoye Qu, Ziyan Liu, Yeqiu Cheng, and Yu Cheng. 2026. Faithrl: Learning to reason faithfully through step-level faithfulness maximization.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, Vienna, Austria. Association for Computational Linguistics.

Jujie He, Tianwen Wei, Rui Yan, Jiacai Liu, Chaojie Wang, Yimeng Gan, Shiwen Tu, Chris Yuhao Liu, Liang Zeng, Xiaokun Wang, Boyang Wang, Yongcong Li, Fuxiang Zhang, Jiacheng Xu, Bo An, Yang Liu, and Yahui Zhou. 2024. Skywork-o1 open series.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jiameng Huang, Baijiong Lin, Guhao Feng, Jierun Chen, Di He, and Lu Hou. 2026. Efficient reasoning for large reasoning language models via certainty-guided reflection suppression. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3ot: generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. 2025. Generator-assistant stepwise rollback framework for large language model agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17683–17700, Suzhou, China. Association for Computational Linguistics.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. 2025. ThinkSwitcher: When to think hard, when to think fast. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5185–5201, Suzhou, China. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. 2025a. Plan and budget: Effective and efficient test-time scaling on large language model reasoning. *Preprint*, arXiv:2505.16122.

Weizhe Lin, Xing Li, Zhiyuan Yang, Xiaojin Fu, Hui-Ling Zhen, Yaoyuan Wang, Xianzhi Yu, Wulong Liu, Xiaosong Li, and Mingxuan Yuan. 2025b. Trimr: Verifier-based training-free thinking compression for efficient test-time scaling. *Preprint*, arXiv:2505.17155.

Wanlong Liu, Junxiao Xu, Fei Yu, Yukang Lin, Ke Ji, Wenyu Chen, Lifeng Shang, Yasheng Wang, Yan Xu, and Benyou Wang. 2025a. QFFT, question-free fine-tuning for adaptive reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. 2025b. Diffadapt: Difficulty-adaptive reasoning for token-efficient llm inference. *Preprint*, arXiv:2510.19669.

Xin Liu and Lu Wang. 2025. Answer convergence as a signal for early stopping in reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17907–17918, Suzhou, China. Association for Computational Linguistics.

Zhipeng Liu, Xuefeng Bai, Kehai Chen, Xinyang Chen, Xiucheng Li, Yang Xiang, Jin Liu, Hong-Dong Li, Yaowei Wang, Liqiang Nie, and Min Zhang. 2025c. A survey on the feedback mechanism of llm-based ai agents. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 10582–10592. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Ruilin Luo, Zhuofan Zheng, Yifan Wang, Yiyao Yu, Xinzhe Ni, Zicheng Lin, Jin Zeng, and Yujiu Yang. 2025. Ursa: Understanding and verifying chain-of-thought reasoning in multimodal mathematics. *arXiv preprint arXiv:2501.04686*.

MAA Committees. 2024. AIME Problems and Solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

MAA Committees. 2025. AIME Problems and Solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions.

OpenAI. 2025. Learning to reason with llms. https://openai.com/research/learning-to-reason-with-llms.

Deng Qiyuan, Xuefeng Bai, Kehai Chen, Yaowei Wang, Liqiang Nie, and Min Zhang. 2025. Efficient safety alignment of large language models via preference re-ranking and representation-based reward modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31156–31171, Vienna, Austria. Association for Computational Linguistics.

Xiaoye Qu, Yafu Li, Zhao-Chen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, Peng Li, Wei Wei, Jing Shao, Chaochao Lu, Yue Zhang, Xian-Sheng Hua, Bowen Zhou, and Yu Cheng. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *Preprint*, arXiv:2503.21614.

QwenTeam. 2025. Qwq-32b: Embracing the power of reinforcement learning.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483.

Zhiwen Ruan, Yixia Li, He Zhu, Yun Chen, Peng Li, Yang Liu, and Guanhua Chen. 2025a. Enhancing large language model reasoning via selective critical token fine-tuning.

Zhiwen Ruan, Yixia Li, He Zhu, Longyue Wang, Weihua Luo, Kaifu Zhang, Yun Chen, and Guanhua Chen. 2025b. LayAlign: Enhancing multilingual reasoning in large language models via layer-wise adaptive fusion and alignment strategy. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1481–1495, Albuquerque, New Mexico. Association for Computational Linguistics.

Swarnadeep Saha, Archiki Prasad, Justin Chen, Peter Hase, Elias Stengel-Eskin, and Mohit Bansal. 2025. System 1.x: Learning to balance fast and slow planning with language models. In *The Thirteenth International Conference on Learning Representations*.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. 2025. DAST: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2322–2331, Suzhou (China). Association for Computational Linguistics.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter L. Bartlett, and Andrea Zanette. 2024. Fast best-of-n decoding via speculative rejection. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. DART-math: Difficulty-aware rejection tuning for mathematical problem-solving. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025a. Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7459–7482, Suzhou, China. Association for Computational Linguistics.

Hongru Wang, Deng Cai, Wanjun Zhong, Shijue Huang, Jeff Z. Pan, Zeming Liu, and Kam-Fai Wong. 2025b. Self-reasoning language models: Unfold hidden reasoning chains with few reasoning catalyst. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5578–5596, Vienna, Austria. Association for Computational Linguistics.

Jingyao Wang, Wenwen Qiang, Zeen Song, Changwen Zheng, and Hui Xiong. 2025c. Learning to think: Information-theoretic reinforcement fine-tuning for LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Ke Wang, Jiahui Zhu, Minjie Ren, Zeming Liu, Shiwei Li, Zongye Zhang, Chenkai Zhang, Xiaoyu Wu, Qiqi Zhan, Qingjie Liu, and Yunhong Wang. 2024a. A survey on data synthesis and augmentation for large language models. *CoRR*, abs/2410.12896.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Weiqin Wang, Yile Wang, Kehao Chen, and Hui Huang. 2025d. Beyond majority voting: Towards fine-grained and more reliable reward signal for test-time reinforcement learning. *arXiv preprint arXiv:2512.15146*.

Weiqin Wang, Yile Wang, and Hui Huang. 2025e. Ranked voting based self-consistency of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14410–14426, Vienna, Austria. Association for Computational Linguistics.

Yi Wang, Junxiao Liu, Shimao Zhang, Jiajun Chen, and Shujian Huang. 2025f. Pats: Process-level adaptive thinking mode switching. *Preprint*, arXiv:2505.19250.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *Preprint*, arXiv:2502.18600.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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

Lei Yang, Shaoyang Xu, Jianxiang Peng, Shaolin Zhu, and Deyi Xiong. 2025c. DCIS: Efficient length extrapolation of LLMs via divide-and-conquer scaling factor search. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15168–15182, Suzhou, China. Association for Computational Linguistics.

Zewei Yu, Lirong Gao, Yuke Zhu, Bo Zheng, Junbo Zhao, Sheng Guo, and Haobo Wang. 2026. Stop unnecessary reflection: Training LRMs for efficient reasoning with adaptive reflection and length coordinated penalty. In *The Fourteenth International Conference on Learning Representations*.

Shaolin Zhu, Supryadi, Shaoyang Xu, Haoran Sun, Leiyu Pan, Menglong Cui, Jiangcun Du, Renren Jin, António Branco, and Deyi Xiong. 2024. Multilingual large language models: A systematic survey. *CoRR*, abs/2411.11072.

A Extended Related Work

A growing line of work mitigates overthinking by post-training LRMs to internalize adaptive reasoning length, typically via variable-length CoT supervision or efficiency-aware alignment signals (Liu et al., 2025a; Kang et al., 2025; Aggarwal and Welleck, 2025; Cheng et al., 2025; Yu et al., 2026). The shared goal is to discourage redundant deliberation on easy instances while retaining sufficient computation for hard problems.

For example, Chen et al. (2025) systematically characterize overthinking in o1-like models and propose a self-training-based mitigation strategy that reduces redundant reasoning without sacrificing accuracy; Shen et al. (2025) introduce difficulty-adaptive slow thinking by calibrating a token-length budget with problem difficulty and performing budget-aware preference optimization, penalizing unnecessarily long CoT on easy questions while encouraging sufficient deliberation on hard ones; Wang et al. (2025c) propose an information-theoretic reinforcement fine-tuning framework with dense process rewards derived from estimated information gain to optimize effective reasoning under limited but sufficient token usage.

Related post-training reasoning optimization also includes off-policy guidance (Yan et al.), step-level faithfulness-aware reinforcement learning (Gui et al., 2026), finer-grained pseudo-label or reward design for test-time reinforcement learning (Wang et al., 2025d), selective critical-token fine-tuning (Ruan et al., 2025a), and catalyst-based self-training that unfolds hidden reasoning chains (Wang et al., 2025b).

Despite their effectiveness, post-training approaches often entail additional optimization per backbone and rely on curated variable-length CoT data or carefully designed efficiency rewards, which can increase training cost and engineering overhead and may risk overfitting to specific output formats. A parallel stream of research addresses these limitations by exploring training-free inference-time interventions (Yang et al., 2025b; Huang et al., 2026), alongside broader feedback-driven and reward-modeling mechanisms in LLM systems (Li et al., 2025; Qiyuan et al., 2025; Liu et al., 2025c). Our work builds upon and extends this direction, aiming for a solution that is both effective and broadly applicable.

Inference-time Aggregation and Broader Efficient Reasoning. Another adjacent line of work improves reasoning through inference-time aggregation and broader efficiency-oriented mechanisms. Ranked-voting self-consistency aggregates ranked answers across multiple samples rather than relying only on final-answer majority voting (Wang et al., 2025e). More broadly, recent survey work systematizes efficient reasoning methods for LRMs across pre-training, post-training, and inference-time stages (Qu et al., 2025). Adjacent efficiency work such as DCIS focuses on context-length extrapolation and reducing long-context adaptation costs, rather than dynamically modulating within-trajectory reasoning depth (Yang et al., 2025c). In contrast, SAT reallocates computation within a single reasoning trajectory on a step-by-step basis.

Data-Centric and Multilingual Perspectives. Broader studies have also highlighted the importance of data generation and multilingual capability for future LLM development. Survey work on data synthesis and augmentation reviews how synthetic and augmented data support LLM construction across the full lifecycle, offering useful context for methods that depend on generated reasoning traces or additional supervision (Wang et al., 2024a). In multilingual settings, the systematic survey on multilingual LLMs and LayAlign highlight challenges in cross-lingual reasoning, alignment, and evaluation (Zhu et al., 2024; Ruan et al., 2025b). Although orthogonal to SAT's current setting, these works suggest promising directions for extending efficient reasoning control to multilingual and data-scarce scenarios.

B Extended Motivation and Case Study

In this section, we elaborate on the motivation behind SAT by analyzing a specific failure mode in complex reasoning tasks, as illustrated in Figure 15.

The example task involves a probability problem requiring the inclusion-exclusion principle: *"Draw 7 cards from a 52-card deck. Find* $P(\geq 1A, \geq 1\heartsuit)$*."* As shown in Figure 15, we compare three approaches:

- **Early-Stopping (Baseline 1):** The model wastes compute performing redundant checks on trivial steps (e.g., verifying basic arithmetic like $13 + 39 = 52$ in Step 1). Crucially, it

Category	Feature Name	Description & Definition
Semantic	Step Embedding	The 384-dimensional dense vector encoded by `gte-small` representing the semantic content of the current reasoning step text.
Uncertainty (Static)	`canonical_logprobs`	The log-probability of the sampled token: $\log P(x_t \mid x_{<t})$.
	`canonical_selected_rank`	The rank of the sampled token in the vocabulary distribution (1-based).
	`canonical_entropy`	The Shannon entropy of the local probability distribution (truncated to Top-K).
	`canonical_logit_gap`	The difference between the largest logit (Top-1) and the second largest logit (Top-2).
	`canonical_margin`	The probability margin between the Top-1 and Top-2 tokens: $P(x_{\text{top1}}) - P(x_{\text{top2}})$.
	`canonical_topk_mass@5`	The cumulative probability mass of the top-5 tokens.
	`canonical_topk_mass@10`	The cumulative probability mass of the top-10 tokens.
Uncertainty (Dynamic)	`canonical_d_logp`	First-order difference of log-probabilities: $\texttt{logp}_t - \texttt{logp}_{t-1}$.
	`canonical_d_entropy`	First-order difference of entropy: $\texttt{ent}_t - \texttt{ent}_{t-1}$.
	`canonical_d_margin`	First-order difference of margin: $\texttt{margin}_t - \texttt{margin}_{t-1}$.
	`canonical_z_logp`	Sliding-window Z-score of log-probability, capturing local anomalies relative to a history window (window size $W_z = 20$).

Table 2: **List of Input Features for the Lightweight Pilot.** The features are categorized into Semantic inputs and Uncertainty inputs (subdivided into Static and Dynamic metrics).

Control Tag	Instruction / Semantics
`[Fast_Step]`	Indicates the current step appears easy. The model is instructed to keep reasoning brief and avoid unnecessary details.
`[Slow_Step]`	Indicates the current step appears difficult. The model is instructed to perform detailed, expansive reasoning.
`[Normal_Step]`	Indicates moderate difficulty. The model is instructed to resume standard step-by-step reasoning. (Corresponds to the NORMAL state).
`[Skip_Step]`	Indicates the step is excessively difficult and further elaboration is unlikely to be helpful. The model is instructed to summarize existing reasoning, make a reasonable guess, and quickly output the final answer.

Table 3: **Definitions of Control Tags.** These tags are inserted into the context to modulate the reasoning style of the subsequent step.

exits the reasoning process too early. After deriving the first-pass answer ($P_{fp} \approx 0.3849$), the model halts without performing a deep verification of the intersection term $P(A_0 \cap H_0)$, leading to an incorrect result.

- **Suppressing Reflective (Baseline 2):** This method aggressively removes all "checking" tokens (e.g., "Wait, let me check...") to maximize speed. While this reduces token usage, it removes the model's ability to self-correct. Consequently, it executes the standard, flawed logic for the inclusion-exclusion step and arrives at the same incorrect probability (0.3849) as the Early-Stopping baseline.

- **Our Method (SAT):** SAT demonstrates dynamic compute allocation.

  1. **Fast Mode:** For Step 1 (Split overlaps) and Step 2 (Inclusion-Exclusion setup), the model identifies the sub-steps as "easy" and skips redundant checks, mimicking the speed of the suppression method.
  2. **Slow Mode:** In Step 4, the model detects the complexity of the verification required for the intersection term. It engages a "Deep verify" process, realizing that the number of Aces in the "Non-Heart" set is 3, not 4 (since the Ace of Hearts is excluded).

  This targeted verification allows SAT to cor-

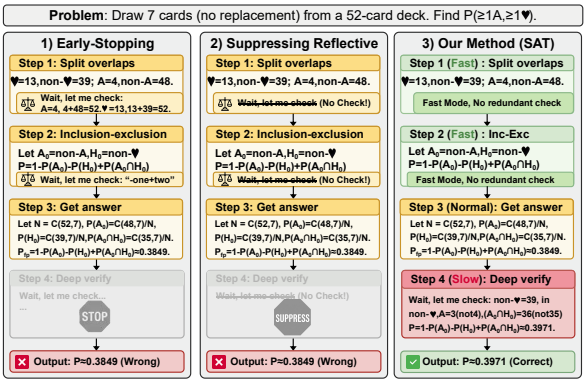

Figure 15: **Baselines vs. SAT.** Early-Stopping halts after a high-confidence first-pass answer and fails; Suppressing Reflective eliminates all verification steps, preventing necessary self-correction and leading to the same error. In contrast, SAT skips redundancy on easy steps but preserves verification on hard steps, achieving the correct answer efficiently.

rect the logic error and derive the true answer ($P \approx 0.3971$), balancing the efficiency of suppression with the rigor of deep reasoning.

C Details of Input Features

As described in Section 3.2, our lightweight Pilot utilizes a dual-channel input representation $\mathbf{z}_t = [\mathbf{h}_{\text{unc}}; \mathbf{h}_{\text{sem}}]$. The Semantic Features ($\mathbf{h}_{\text{sem}}$) are dense vector representations of the reasoning step's textual content, while the Uncertainty Features ($\mathbf{h}_{\text{unc}}$) are statistical metrics derived from the LLM's decoding logits. Table 2 provides a detailed categorization of these features based on our implementation.

Feature Extraction Process. For every token generated within a reasoning step, we extract a set of raw confidence metrics (e.g., LogProb, Entropy) and dynamic metrics (e.g., Z-score, First-order differences). Specifically, for a reasoning step y_t consisting of multiple tokens, we aggregate these token-level metrics (via mean-pooling) to form the step-level feature vector $\mathbf{h}_{\text{unc}}$. The semantic vector $\mathbf{h}_{\text{sem}}$ is obtained by encoding the raw text of the step using the frozen `gte-small` encoder.

D Control Tags and System Prompts

To enable dynamic control over the reasoning depth, we inject specific control tags into the model's context based on the state determined by the FSM. These tags are defined in the system prompt to guide the model's behavior via instruction following.

Control Tags. Table 3 details the semantics of each tag used in our framework. Note that while the FSM defines a NORMAL state, we map this to the tag `[Normal_Step]` in the prompt to explicitly signal a return to moderate reasoning depth.

System Prompt. The full system prompt used to initialize the model and define these behaviors is provided below. This prompt is prepended to the user query. For code-generation benchmarks (e.g., HumanEval), we use a separate system prompt that requests the model to output only the final Python implementation within "`python code blocks`", to avoid format interference from `\boxed{}`.

System Prompt for In-Context Steering

```
Please reason step by step, and
put your final answer within
\boxed{}.

During your thinking, you may see
the following tags:
 • [Fast_Step] means the current
   step seems easy; please keep
   your reasoning brief and avoid
   unnecessary details.
 • [Slow_Step] means the current
   step seems difficult; please
   perform detailed reasoning.
 • [Normal_Step] means the
   current step has moderate
   difficulty; please resume normal
   step-by-step reasoning.
 • [Skip_Step] means this step
   is too difficult and further
   detailed expansion is not
   very helpful, please summarize
   the existing reasoning, make
   a reasonable guess for the
   conclusion, and then quickly
   output the final answer.

{question}
```

E Implementation Details

E.1 Experimental Environment

All experiments are conducted on **NVIDIA H20 (96GB)** GPUs using the **HuggingFace Transformers** library. Following official recommendations, we set sampling parameters to Temperature $= 0.6$ and Top-p $= 0.95$, with a maximum generation length limit of 16,384 tokens. For SAT on AMC, AIME 2024, and AIME 2025, we report the average over three runs due to the small benchmark size.

E.2 Hyperparameter Settings

For SAT, the detailed FSM configurations are:

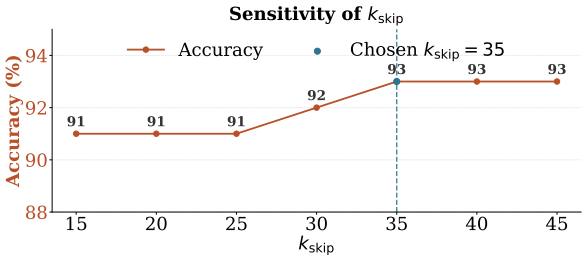

Figure 16: **Sensitivity of $k_{\texttt{skip}}$ on the first 100 MATH500 problems using Qwen3-8B.** Smaller values (15–25) are less stable and reduce accuracy, while performance saturates once $k_{\texttt{skip}} \geq 35$.

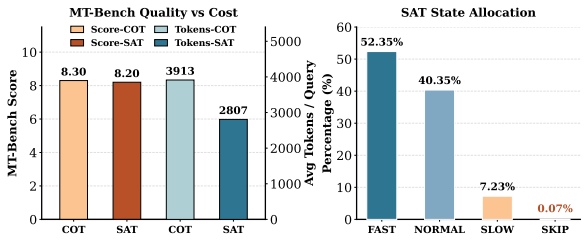

Figure 17: **MT-Bench evaluation on Qwen3-8B.** Left: SAT preserves comparable dialogue quality while reducing generation cost. Right: SAT allocates computation conservatively in open-ended dialogue, with most steps staying in FAST or NORMAL modes.

- **Thresholds:** $\tau_{\texttt{fast}} = 0.2$, $\tau_{\texttt{slow}} = 0.6$, $\tau_{\texttt{skip}} = 0.85$.
- **Hysteresis Margin:** $\Delta = 0.1$ (to prevent state flickering).
- **History Windows:** $k_{\texttt{fast}} = 6$, $k_{\texttt{slow}} = 5$, $k_{\texttt{skip}} = 35$.

Rationale for history-window selection. The history windows are chosen to balance responsiveness and stability. We use relatively short windows for $k_{\texttt{fast}}$ and $k_{\texttt{slow}}$ so that the FSM can react to short-term local difficulty changes without excessive lag. In contrast, $k_{\texttt{skip}}$ is intentionally more conservative, since SKIP acts as a safety fallback and should only be triggered when the model exhibits sustained high difficulty or low progress, rather than a transient noisy spike.

$k_{\texttt{skip}}$ sensitivity. We further analyze $k_{\texttt{skip}}$ on the first 100 MATH500 problems using Qwen3-8B. As shown in Figure 16, smaller values (15–25) yield only 91% accuracy, indicating a higher risk of premature SKIP activation. Accuracy rises to 92% at $k_{\texttt{skip}}$=30 and stabilizes at 93% once $k_{\texttt{skip}} \geq 35$, suggesting that $k_{\texttt{skip}}$=35 provides a robust balance between avoiding early intervention and maintaining stable control behavior.

E.3 Baseline Reproduction

For baselines DEER, CGRS, and ThinkSwitcher, we prioritize citing reported data from their original papers, while results for NoThinking, TALE, and Dynasor are sourced from the CGRS paper (Huang et al., 2026).

F Additional Experimental Analysis

F.1 Comparisons with Additional Baselines

Due to space limitations in the main text, we present comparisons with three additional baseline strategies in this section to provide a broader context for SAT's performance. These methods represent distinct approaches to the efficiency-accuracy trade-off:

- **NoThinking**: A direct prompting baseline that instructs the model to bypass intermediate reasoning processes entirely and generate the final answer directly. This method prioritizes maximum efficiency by eliminating the Chain-of-Thought.
- **TALE** (Han et al., 2025): A constraint-based prompting strategy that explicitly instructs the model to solve the problem within a pre-defined token budget. By imposing a strict length constraint on the generation process, TALE encourages the model to condense its reasoning steps and prioritize essential information, thereby reducing computational costs while attempting to maintain reasoning integrity.
- **Dynasor** (Fu et al., 2025): A decoding manipulation method that periodically extracts intermediate answers at fixed token intervals during generation. It employs an early-exit mechanism, terminating the inference process if multiple consecutive checks yield consistent answers, thereby reducing redundant computation.

The results are detailed in Table 4 (below).

F.2 Open-Ended Dialogue Evaluation on MT-Bench

We further evaluate SAT on MT-Bench to assess its behavior in open-ended dialogue. As shown in Figure 17, SAT achieves a comparable MT-Bench score to COT (8.20 vs. 8.30) while reducing the average generated tokens per query from 3912.79 to 2806.71 (-28.3%). The state distribution further

Model	Method	GSM8K		MATH500		AIME 2024		AIME 2025		AMC	
		Acc (↑, %)	Tokens (↓)	Acc (↑, %)	Tokens (↓)	Acc (↑, %)	Tokens (↓)	Acc (↑, %)	Tokens (↓)	Acc (↑, %)	Tokens (↓)
Qwen3-4B	COT	**95.1**	2136	95.0	4823	60.0	12662	50.0	12720	92.5	7408
	TALE	–	–	89.1 (-5.9)	2657 (-45%)	48.9 (-11.1)	9727 (-23%)	–	–	86.7 (-5.8)	5107 (-31%)
	NoThinking	–	–	84.9 (-10.1)	**988 (-80%)**	24.4 (-35.6)	**4504 (-64%)**	–	–	70.0 (-22.5)	**1710 (-77%)**
	Dynasor	–	–	90.1 (-4.9)	3877 (-20%)	54.3 (-5.7)	9912 (-22%)	–	–	86.7 (-5.8)	6233 (-16%)
	DEER	94.5 (-0.6)	1250 (-41%)	92.6 (-2.4)	3214 (-33%)	**63.3 (+3.3)**	9327 (-26%)	55.0 (+5.0)	12084 (-5%)	87.5 (-5.0)	4906 (-34%)
	CGRS	–	–	91.3 (-3.7)	2704 (-44%)	56.7 (-3.3)	7893 (-38%)	–	–	86.7 (-5.8)	4351 (-41%)
	SAT	**95.1 (+0.0)**	**845 (-60%)**	**95.6 (+0.6)**	2833 (-41%)	61.1 (+1.1)	9184 (-27%)	**56.6 (+6.6)**	**10228 (-20%)**	**93.3 (+0.8)**	5145 (-31%)
Qwen3-8B	COT	**95.8**	2152	95.6	5166	66.7	12393	**60.0**	12835	90.0	7920
	TALE	–	–	92.3 (-3.3)	3885 (-25%)	**68.9 (+2.2)**	10942 (-12%)	–	–	88.3 (-1.7)	6872 (-13%)
	NoThinking	–	–	87.1 (-8.5)	**1239 (-76%)**	30.0 (-36.7)	**5967 (-52%)**	–	–	72.5 (-17.5)	**2426 (-69%)**
	Dynasor	–	–	91.7 (-3.9)	3841 (-26%)	62.2 (-4.5)	10174 (-18%)	–	–	89.2 (-0.8)	6457 (-18%)
	DEER	95.5 (-0.3)	981 (-54%)	92.6 (-3.0)	2732 (-47%)	61.7 (-5.0)	8796 (-29%)	**60.0 (+0.0)**	12229 (-5%)	**92.5 (+2.5)**	4392 (-45%)
	CGRS	–	–	93.3 (-2.3)	3507 (-32%)	61.1 (-5.6)	8792 (-29%)	–	–	89.2 (-0.8)	5595 (-29%)
	SAT	95.5 (-0.3)	**879 (-59%)**	**95.8 (+0.2)**	3215 (-38%)	**68.9 (+2.2)**	9281 (-25%)	58.9 (-1.1)	**10909 (-15%)**	90.0 (+0.0)	5062 (-36%)
Qwen3-14B	COT	95.7	1660	96.2	4598	73.3	11742	56.7	12820	92.5	6964
	TALE	–	–	93.7 (-2.5)	3389 (-26%)	71.1 (-2.2)	10860 (-8%)	–	–	92.5 (+0.0)	5951 (-15%)
	NoThinking	–	–	87.0 (-9.2)	**853 (-81%)**	27.8 (-45.5)	**3689 (-69%)**	–	–	77.5 (-15.0)	**1616 (-77%)**
	Dynasor	–	–	84.4 (-11.8)	3667 (-20%)	65.6 (-7.7)	9775 (-17%)	–	–	90.0 (-2.5)	6030 (-13%)
	DEER	95.3 (-0.4)	840 (-49%)	94.0 (-2.2)	3074 (-33%)	**76.7 (+3.4)**	7619 (-35%)	**66.7 (+10.0)**	11135 (-13%)	95.0 (+2.5)	4763 (-32%)
	CGRS	–	–	94.5 (-1.7)	3235 (-30%)	70.0 (-3.3)	8662 (-26%)	–	–	93.3 (+0.8)	5076 (-27%)
	SAT	**96.4 (+0.7)**	**767 (-54%)**	**96.6 (+0.4)**	2904 (-37%)	71.1 (-2.2)	9598 (-18%)	61.1 (+4.4)	**10630 (-17%)**	**95.8 (+3.3)**	4835 (-31%)
Qwen3-32B	COT	96.0	1688	95.6	4358	70.0	10788	63.3	12203	95.0	6448
	TALE	–	–	93.6 (-2.0)	3857 (-11%)	67.8 (-2.2)	10688 (-1%)	–	–	93.3 (-1.7)	6533 (+1%)
	NoThinking	–	–	87.0 (-8.6)	**1054 (-76%)**	41.1 (-28.9)	**5635 (-48%)**	–	–	75.0 (-20.0)	**2221 (-66%)**
	Dynasor	–	–	85.2 (-10.4)	3486 (-20%)	64.4 (-5.6)	9518 (-12%)	–	–	92.5 (-2.5)	5521 (-14%)
	DEER	96.2 (+0.2)	769 (-54%)	94.2 (-1.4)	3418 (-22%)	**76.7 (+6.7)**	8682 (-20%)	66.7 (+3.4)	10893 (-11%)	**97.5 (+2.5)**	5753 (-11%)
	CGRS	–	–	93.1 (-2.5)	2993 (-31%)	65.6 (-4.4)	8128 (-25%)	–	–	94.2 (-0.8)	4766 (-26%)
	SAT	**96.5 (+0.5)**	**680 (-60%)**	**96.6 (+1.0)**	2719 (-38%)	70.0 (+0.0)	9306 (-14%)	**70.0 (+6.7)**	**10011 (-18%)**	95.8 (+0.8)	4911 (-24%)
DS-Qwen-1.5B	COT	77.6	1193	82.2	4723	**26.7**	11941	23.3	11879	67.5	7729
	DEER	74.7 (-2.9)	984 (-18%)	67.8 (-14.4)	**2497 (-47%)**	23.3 (-3.4)	9553 (-20%)	10.0 (-13.3)	9281 (-22%)	60.0 (-7.5)	**5496 (-29%)**
	ThinkSwitcher	**84.7** (+7.1)	2114 (+77%)	**82.4 (+0.2)**	4544 (-4%)	23.3 (-3.4)	**8192 (-31%)**	**28.3 (+5.0)**	**6689 (-43%)**	–	–
	SAT	77.3 (-0.3)	**564 (-53%)**	**82.4 (+0.2)**	3230 (-32%)	**26.7 (+0.0)**	10893 (-9%)	20.0 (-3.3)	9455 (-20%)	**68.3 (+0.8)**	6642 (-14%)
DS-Qwen-7B	COT	89.9	532	**92.8**	3537	50.0	12662	30.0	11028	87.5	6366
	TALE	–	–	89.1 (-3.7)	2657 (-25%)	48.9 (-1.1)	9727 (-23%)	–	–	86.7 (-0.8)	5107 (-20%)
	NoThinking	–	–	80.9 (-11.9)	**1173 (-67%)**	32.2 (-17.8)	**6680 (-47%)**	–	–	75.8 (-11.7)	**2499 (-61%)**
	Dynasor	–	–	81.8 (-11.0)	2070 (-41%)	47.8 (-2.2)	8334 (-34%)	–	–	84.2 (-3.3)	5201 (-18%)
	DEER	90.6 (+0.7)	917 (+72%)	89.8 (-3.0)	2143 (-39%)	49.2 (-0.8)	9839 (-22%)	36.7 (+6.7)	7257 (-34%)	85.0 (-2.5)	4451 (-30%)
	CGRS	–	–	87.6 (-5.2)	1867 (-47%)	**52.2 (+2.2)**	7597 (-40%)	–	–	**88.3 (+0.8)**	3406 (-46%)
	ThinkSwitcher	**92.5 (+2.6)**	1389 (+161%)	91.3 (-1.5)	3495 (-1.0%)	48.3 (-1.7)	7936 (-37%)	**37.5 (+7.5)**	**6948 (-37%)**	–	–
	SAT	89.3 (-0.6)	**385 (-28%)**	**92.8 (+0.0)**	2237 (-37%)	51.1 (+1.1)	8336 (-34%)	36.7 (+6.7)	9496 (-14%)	**88.3 (+0.8)**	3642 (-43%)
DS-Qwen-14B	COT	94.9	1122	94.4	3539	60.0	10343	36.7	11002	92.5	5333
	DEER	93.3 (-1.6)	1040 (-7%)	89.8 (-4.6)	2577 (-27%)	**68.4 (+8.4)**	8115 (-22%)	36.7 (+0.0)	10125 (-8%)	85.0 (-7.5)	**4240 (-20%)**
	ThinkSwitcher	94.3 (-0.6)	1042 (-7%)	92.7 (-1.7)	3572 (+1%)	60.4 (+0.4)	**8044 (-22%)**	**42.5 (+5.8)**	10065 (-9%)	–	–
	SAT	**95.2 (+0.3)**	**621 (-45%)**	**94.8 (+0.4)**	**2515 (-29%)**	62.2 (+2.2)	8413 (-19%)	38.9 (+2.2)	**8912 (-19%)**	**93.3 (+0.8)**	4582 (-14%)
Nemo-Llama-8b	COT	**90.2**	1400	93.8	3412	56.7	10280	**40**	10893	90	5997
	DEER	89.8 (-0.4)	1473 (+5%)	91.4 (-2.4)	2995 (-12%)	**66.7 (+10)**	9755 (-5%)	36.7 (-3.3)	11820 (+9%)	90 (+0.0)	5408 (-10%)
	SAT	89.7 (-0.5)	**1035 (-26%)**	**94 (+0.2)**	**2844 (-17%)**	57.8 (+1.1)	**9065 (-12%)**	35.6 (-4.4)	**10555 (-3%)**	**95.8 (+5.8)**	**4610 (-23%)**
QwQ-32B	COT	**97.0**	1561	**97.0**	4025	66.7	11305	**63.3**	12554	90.0	7086
	TALE	–	–	94.0 (-3.0)	3533 (-12%)	61.1 (-5.6)	10888 (-4%)	–	–	90.8 (+0.8)	6522 (-8%)
	NoThinking	–	–	94.2 (-2.8)	4276 (+6%)	62.2 (-4.5)	11688 (+3%)	–	–	88.3 (-1.7)	7493 (+6%)
	Dynasor	–	–	94.0 (-3.0)	4156 (+3%)	64.4 (-2.3)	9733 (-14%)	–	–	90.0 (+0.0)	7185 (+1%)
	DEER	96.3 (-0.7)	977 (-37%)	94.6 (-2.4)	3316 (-18%)	70.0 (+3.3)	10097 (-11%)	50.0 (-13.3)	11598 (-8%)	**95.0 (+5.0)**	5782 (-18%)
	CGRS	–	–	94.2 (-2.8)	**2810 (-30%)**	68.9 (-1.1)	**8202 (-27%)**	–	–	93.3 (+3.3)	**4771 (-33%)**
	SAT	96.6 (-0.4)	**969 (-38%)**	**97.0 (+0.0)**	3256 (-19%)	**71.1 (+4.4)**	9288 (-18%)	58.9 (-4.4)	**10849 (-14%)**	94.2 (+4.2)	6053 (-15%)

Table 4: **Comparisons with additional baselines.** Metrics include **Acc** (↑) and **Tokens** (↓). Changes relative to the COT are highlighted in orange for Acc and blue for Tokens. Best results within each group are bolded.

shows that SAT allocates computation on demand in dialogue scenarios: FAST and NORMAL account for 52.35% and 40.35% of decisions, respectively, while only 7.23% enter SLOW and SKIP is nearly unused (0.07%). These results suggest that SAT can suppress redundant deliberation beyond closed-form reasoning benchmarks and remain effective in open-ended interaction settings.

F.3 Stress Test on Qwen3-0.6B

To further probe the lower end of model scale, we evaluate SAT on Qwen3-0.6B over four mathematical benchmarks. As shown in Figure 18, SAT consistently reduces token usage on all datasets, with reductions of 25.6% on MATH500, 21.6% on AMC, 15.3% on AIME 2024, and 8.0% on AIME 2025. At the same time, it preserves or improves accuracy on three of the four benchmarks, while the only degradation appears on AIME 2025 (-3.3 points). This suggests that SAT remains applicable

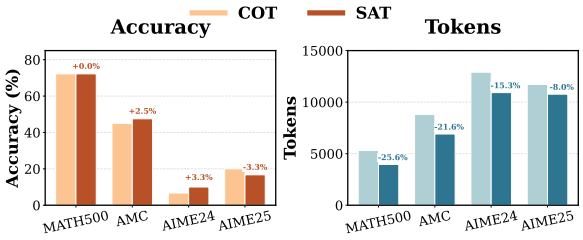

Figure 18: **Stress test on Qwen3-0.6B.** SAT remains effective on this sub-1B backbone, matching or improving accuracy on three of four mathematical benchmarks while consistently reducing tokens.

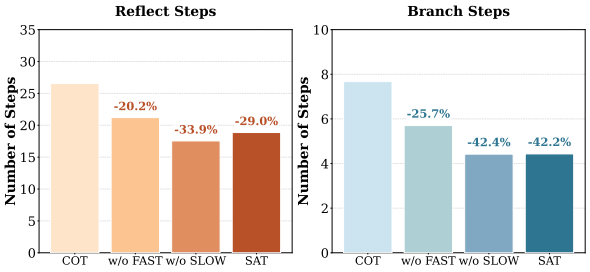

Figure 19: **Process behavior analysis on MATH500.** Average `reflect_steps` and `branch_steps` (lower is better) for *COT*, *w/o FAST*, *w/o SLOW*, and SAT, computed from the thought traces using the forward window scheme (W=1).

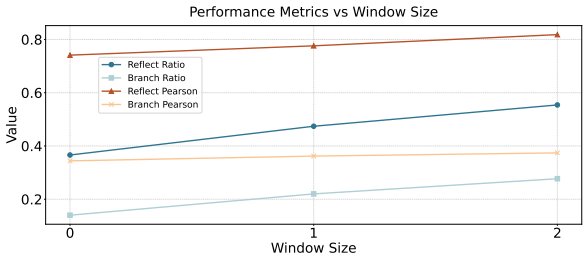

Figure 20: **Window-size sensitivity for token-savings attribution.** We report (i) contribution ratios of reflection-/branch-related savings and (ii) Pearson correlations between per-sample total savings and each component, under window sizes $W \in \{0, 1, 2\}$.

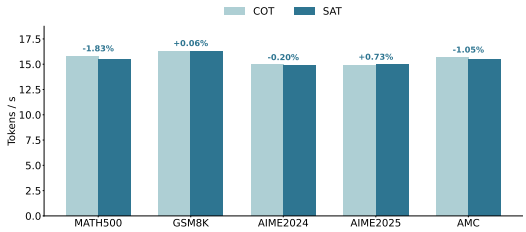

Figure 21: **Generation Throughput Comparison.** The average number of tokens generated per second by COT and SAT using Qwen3-32B. The consistent throughput confirms that SAT introduces negligible per-step latency overhead.

even below the 1B scale.

F.4 Process-Level Behavior Analysis on MATH500

Definition of reflection/branching cues. We analyze the *thought part* (text before `</think>`) and split it into step units by newline delimiters. A *reflection cue* is a lexical indicator that the model is verifying, reconsidering, or correcting its reasoning. Typical examples include phrases such as *"check"*, *"verify"*, *"wait"*, or *"actually"*. Similarly, a *branching cue* indicates explicit exploration of alternatives, e.g., *"case 1/2"*, *"another approach"*, or *"on the other hand"*. We use a fixed cue list shared across all methods for fairness.

Computation of `reflect_steps` and `branch_steps`. We first mark a step as a cue-hit if it contains any cue term. Under the forward window scheme (default W=1), if step i is a cue-hit, then steps i through $i{+}W$ are marked as belonging to a reflection (or branching) segment. `reflect_steps` and `branch_steps` are defined as the total number of marked steps for reflection and branching, respectively.

Results. Figure 19 shows that both SAT and the ablated variants reduce reflection and branching behaviors compared to COT. COT exhibits 26.55 `reflect_steps` and 7.67 `branch_steps` on average. *w/o FAST* reduces `reflect_steps` to 21.20 (-20.2%) and `branch_steps` to 5.70 (-25.7%). *w/o SLOW* yields the strongest suppression (reflect: 17.54, -33.9%; branch: 4.42, -42.4%). The full SAT achieves comparable reductions (reflect: 18.86, -29.0%; branch: 4.43, -42.2%), indicating that stepwise mode navigation can effectively curb redundant reflection and excessive branching.

F.5 Sensitivity of attribution metrics to window size

How the attribution numbers are computed. We attribute SAT's token savings to two process behaviors, *reflection* and *branching*, computed on the thought trace (text before `</think>`) split into step units by newline. We first detect *cue steps* using a fixed lexical list shared across all methods (e.g., reflection cues such as "`check`", "`verify`", "`wait`", "`actually`"). To better capture short cue-triggered spans, we apply a forward window expansion: if step i contains a reflection cue, then steps $i, \ldots, i{+}W$ are marked as reflection-related, and the reflection token count is the sum of tokens over marked steps. For a pair of methods (COT vs. SAT), we define the *reflection contribution ratio* as the fraction of total token savings (computed in a positive-only manner to avoid cancellation) that can be explained by the reduction in reflection-marked tokens; we further compute the Pearson correlation between per-sample total savings and per-sample reflection-marked savings. Branch-related ratios and correlations are computed analogously, using branching cues (e.g., "`case`", "`alternative`", "`another way`").

Why we choose $W{=}2$, and robustness. We use $W{=}2$ in the main text because it provides a slightly more inclusive segmentation of cue-triggered spans, yielding a clearer attribution signal while remaining conservative (only a short local expansion). As shown in Figure 20, the qualitative conclusions are stable for smaller windows: even with $W{=}1$, reflection remains the dominant contributor (reflect_ratio 0.474 vs. branch_ratio 0.220) and is strongly aligned with total savings (reflect Pearson 0.776 vs. branch Pearson 0.362). Increasing to $W{=}2$ strengthens the same trend (reflect_ratio 0.554, reflect Pearson 0.818), supporting the interpretation that reduced reflection is the primary driver of SAT's token efficiency, with branching reduction as a secondary factor.

F.6 Throughput Analysis

To rigorously assess the computational overhead introduced by the external Pilot and FSM logic, we compare the generation throughput (measured in tokens per second) of SAT against vanilla COT. The evaluation is conducted using **Qwen3-32B** across five mathematical benchmarks: MATH500, GSM8K, AIME 2024, AIME 2025, and AMC.

As illustrated in Figure 21, the generation speed of SAT is virtually indistinguishable from that of vanilla COT across all datasets (e.g., **15.51** vs. **15.68** tokens/s on AMC; **16.30** vs. **16.30** tokens/s on GSM8K). This parity indicates that the lightweight 30M Pilot operates with negligible latency, likely fully masked by the memory-bound decoding process of the 32B backbone. Consequently, the end-to-end speedup reported in the main text is purely derived from the reduction in total generated tokens, validating that SAT incurs no throughput penalty during inference.

F.7 Setup for Pilot Cross-domain Generalization

Following prior work, we construct a sampling-based difficulty proxy on GPQA and HumanEval by drawing five independent solutions from Qwen3-8B for each problem. Problems are assigned to difficulty levels 0, 1, and 2 according to 0, 1–3, and 4–5 failed samples, respectively. We then average the step-level Pilot scores over the full trajectory to obtain a problem-level score, and compute its correlation with this proxy difficulty.

G Potential Risks

While our proposed framework, Stepwise Adaptive Thinking (SAT), significantly improves the inference efficiency of Large Reasoning Models (LRMs), we acknowledge the broader risks inherent to AI systems that deploy such techniques:

Reliability and Hallucination. Although our method aims to preserve accuracy by retaining verification steps for complex problems, LRMs are fundamentally probabilistic and prone to hallucinations. The dynamic pruning of reasoning steps (e.g., in *Fast* or *Skip* modes) introduces a trade-off where subtle, necessary self-corrections might occasionally be bypassed, potentially leading to confident but incorrect outputs in out-of-distribution scenarios.

Misuse of Efficient Inference. By reducing the computational barrier and token cost of complex reasoning (achieving up to 40% reduction), our work unintentionally lowers the cost for malicious actors to deploy high-capability models at scale. This could facilitate the automated generation of sophisticated disinformation, social engineering attacks, or malicious code at a lower economic cost. We emphasize that human oversight remains critical, especially in high-stakes domains, to ensure that the pursuit of efficiency does not compromise the safety and ethical integrity of the reasoning process.

H Use of AI Assistants

AI assistants were used in a limited and supportive role during the writing process, primarily for language polishing and text refinement.