Mitigating Overthinking via Interleaved System-1/2 Reasoning in Large Language Models

Anonymous ACL submission

Abstract

Large reasoning models improve problem-solving performance by generating extended Chain-of-Thought, but often suffer from overthinking, producing unnecessarily long and redundant reasoning traces. Prior works mainly focus on directly compressing reasoning length, or selecting reasoning modes globally based on task difficulty, without accounting for the dynamic and heterogeneous nature of the reasoning process itself. This lack of consideration for inference-time diversity limits their ability to adapt reasoning strategies flexibly. To address this gap, we propose Interleaved System-1/2 Reasoning **(InterSR)**, a cognitively inspired framework that dynamically alternates between System 1 (fast) and System 2 (slow) reasoning modes during generation. Central to InterSR is a learnable system switching signal that reflects the necessity for deeper reasoning at each intermediate step and enables fine-grained control over cognitive mode transitions. Experiments across mathematical and cross-domain benchmarks show that InterSR effectively captures internal reasoning signals to reduce token usage while maintaining strong and robust accuracy. We release our code at InterSR[1].

1 Introduction

Large Reasoning Model (LRM) has demonstrated a significant performance improvement over instruct models by introducing long Chain-of-Thought (CoT) during test-time scaling to enhance the model's problem-solving ability (Guo et al., 2025; Jaech et al., 2024; Yang et al., 2025a). This extended CoT allows the model to thoroughly consider the problem, decompose complex issues, explore various possibilities, and validate the correctness of its outputs (Feng et al., 2023; Wei et al., 2022; Sprague et al., 2024). However, this long CoT has been found unsuitable for all types of problems. For simple problems (such as "2+3"), long CoT thinking is regarded as an overthinking phenomenon (Chen et al., 2025c; Cuadron et al., 2025; Ghosal et al., 2025; Zhou et al., 2026). This phenomenon is primarily characterized by the model repeatedly verifying the correct answer or engaging in meaningless reasoning (Fu et al., 2025; Wang et al., 2025; Ballon et al., 2025; Chen et al., 2026a). This not only reduces computational efficiency and leads to resource waste, but may also negatively impact the response accuracy (Du et al., 2026).

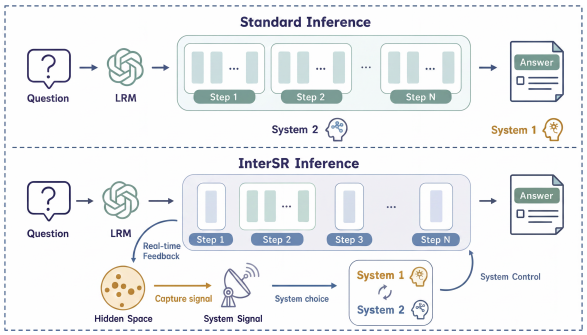

Figure 1: Illustration of InterSR. Standard reasoning applies uniform System 2 (slow) thinking throughout, leading to overthinking on simple steps. InterSR dynamically switches between System 1 (fast) and System 2 reasoning based on step-level complexity, reducing redundant computation while preserving accuracy.

Recent works have proposed several approaches to mitigate overthinking (Feng et al., 2025; Sui et al., 2025; Su et al., 2025; Bin et al., 2026). **Response-side** methods reduce token usage by compressing or truncating the reasoning output (Chen et al., 2025c; Fu et al., 2025; Yang et al., 2025b; Hou et al., 2025), but risk removing critical reasoning steps. **Inference-side** methods classify problems by difficulty and select reasoning modes accordingly (Pan et al., 2024; Qi et al., 2024; Yao et al., 2024), but oversimplify complexity by treating each problem uniformly—ignoring that indi-

[1] https://anonymous.4open.science/r/InterSR-728E

vidual reasoning steps within the same problem can vary dramatically in difficulty. These coarse-grained approaches either sacrifice reasoning quality for efficiency or fail to capture the fine-grained heterogeneity within a single problem, ultimately limiting both effective token savings and stable accuracy preservation (Yu et al., 2025).

Considering these limitations, we propose InterSR, an interleaved system reasoning approach that addresses overthinking at step-level granularity. We model the problem as minimizing computational cost while preserving information gain at each reasoning step. Using semantic entropy (Kuhn et al., 2023; Cheng et al., 2025) as a proxy for information gain, we construct a step-aware system signal that enables dynamic switching between fast and slow reasoning modes at each step.

Our contributions are threefold: (1) We propose InterSR, which dynamically switches between reasoning systems based on a constructed system signal to improve inference allocation and reduce overthinking. (2) We demonstrate that this signal is encoded in the model's hidden states, and train a lightweight predictor to enable near-zero-cost switching during generation. (3) We evaluate InterSR on four math benchmarks, achieving up to 51% token reduction without accuracy loss, and further validate its generalizability across STEM, knowledge-intensive, and code task categories.

2 Background

2.1 Overthinking

Despite the benefits of explicit reasoning, large language models (LLMs) often produce unnecessarily long or redundant chains of thought, even for tasks that can be solved with minimal or no reasoning. This phenomenon, known as **overthinking** (Chen et al., 2025c; Sui et al., 2025), reflects inefficient allocation of inference resources and is commonly observed across many CoT-style models (Pan et al., 2024; Zhang et al., 2025). Overthinking in large language models (LLMs) refers to the production of unnecessarily long reasoning traces that do not substantially improve task performance. It can be formalized as follows:

$$\exists\, r_{\text{short}} \in \mathcal{R} \text{ s.t. } \begin{cases} P(y \mid x, r_{\text{long}}) \approx P(y \mid x, r_{\text{short}}), \\ \mathcal{C}(r_{\text{long}}) \gg \mathcal{C}(r_{\text{short}}) \end{cases} \tag{1}$$

where r_{long} denotes an over-elaborate reasoning sequence, while r_{short} is a more efficient alternative that achieves comparable predictive performance. $\mathcal{C}(\cdot)$ measures reasoning cost, such as token usage or inference latency. This inefficiency typically arises when the model continues generating reasoning content with low marginal utility, for example generating unnecessary intermediate steps.

2.2 Dual-System Theory

Dual-System Theory (Evans, 2017; Chaiken and Trope, 1999) posits that human reasoning operates through two distinct but interacting systems: **System 1** (fast, intuitive) and **System 2** (slow, deliberative). These systems interact dynamically, with System 2 overriding System 1 when deeper reasoning is required for complex decisions (Shi et al., 2025; Shulman et al., 2016).

Recent LLM reasoning research has adopted this framework (Li et al., 2025; Saha et al., 2024; Ziabari et al., 2025; Coda-Forno et al., 2025), mapping System 1 to efficient low-cost inference and System 2 to resource-intensive deliberation (Chen et al., 2025b,a). However, existing methods typically select reasoning modes globally based on overall task difficulty (Qi et al., 2024; Zhao et al., 2025), neglecting the compositional nature of reasoning, where different intermediate steps within the same problem demand varying cognitive effort.

3 Method

In this section, we present the construction process of InterSR. We start by formally modeling the overthinking problem and emphasize the key role of information gain. Then, we introduce a semantic-entropy-based system signal and reveal its relationship with internal hidden states. Building upon these findings, we develop the complete Interleaved System-1/2 Reasoning Framework.

3.1 Stepwise Modeling of the Reasoning Process

We view the reasoning process as a sequence of explicit steps $r = \{r_1, r_2, \ldots, r_T\}$, where each step r_t is dynamically governed by a reasoning mode $m_t \in \{\mathrm{S}_1, \mathrm{S}_2\}$. System 1 provides fast, low-cost, heuristic reasoning, while System 2 offers slow, deliberative, high-cost inference. This decomposition allows each step to flexibly adopt the cognitive mode suited to its semantic demands, enabling fine-grained control over the reasoning trajectory. Unlike prior approaches that statically assign a single system to the entire process, we

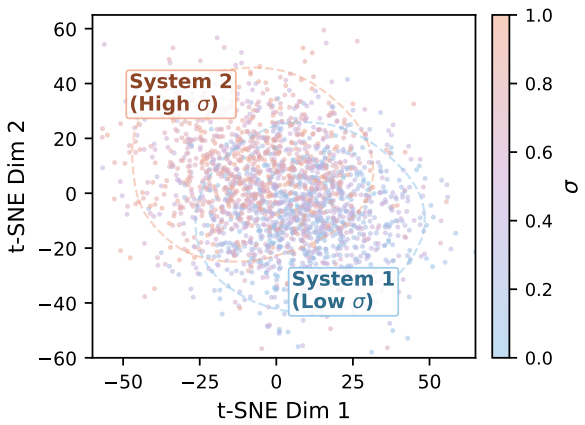

Figure 2: t-SNE projection of hidden representations from Deepseek-R1-distill-Qwen2.5-14B, with points colored by system signal strength. Clear separation between System 1 (low σ) and System 2 (high σ) clusters is observed.

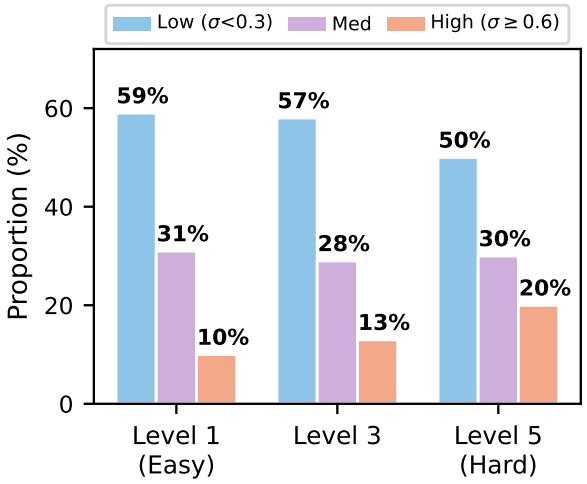

Figure 3: Signal strength distribution on DART-Math by problem difficulty (R1-14B). Harder problems exhibit a higher proportion of High-signal steps, indicating greater System 2 demand.

model reasoning as an interleaved sequence of fast and slow modes. The model generates each step as $r_t \sim P_\theta(r_t \mid x, r_{<t}, m_t)$, where the mode sequence $\{m_t\}$ reflects an adaptive cognitive schedule over time. We assume that, all else being equal, allocating more token resources to a reasoning step can increase the step's epistemic gain. Under this assumption, we invoke System 2 only when necessary to maintain semantic precision while minimizing total computation. This idea can be expressed as the following constrained optimization:

$$\min_{\{z_t\}} \sum_{t=1}^{T} \mathcal{C}(z_t)$$
$$\text{subject to} \begin{cases} \sum_{t=1}^{T} \mathcal{G}(s_t, z_t) = \sum_{t=1}^{T} \mathcal{G}(s_t, 1), \\ z_t \in \{0,1\}, \quad t = 1, \ldots, T \end{cases} \tag{2}$$

where z_t denotes system choice (0 for S_1, 1 for S_2), $\mathcal{C}(z_t)$ represents system cost, and $\mathcal{G}(s_t, z_t)$ captures semantic gain. The goal is to match the epistemic quality of full System 2 reasoning while reducing overall cost through selective system switching.

3.2 Constructing Signals Based on Semantic Entropy

While System 1 is effective when the semantic exploration space of a reasoning step is relatively simple, it fails to extract full epistemic value once the underlying inference space becomes sufficiently complex or ambiguous. In contrast, System 2, though more computationally expensive, can better leverage such latent structure and thus achieve higher information gain. Based on the above modeling, System 1 can efficiently process reasoning steps within simple inference spaces, but fails to capture full epistemic value when a step involves complex or ambiguous semantic structures. To maintain quality equivalent to full System 2 reasoning, we switch to System 2 whenever System 1 cannot extract sufficient information:

$$z_t = \begin{cases} 0, & \text{if } \mathcal{G}(s_t, 0) \geq T, \\ 1, & \text{otherwise}, \end{cases} \tag{3}$$

where $z_t \in \{0, 1\}$ indicates the system choice at step t (0 for System 1, 1 for System 2), $\mathcal{G}(s_t, 0)$ measures the semantic gain attainable by System 1 at step s_t, and T is a threshold defining sufficient gain. When the gain under System 1 falls below T, System 2 is invoked to ensure comprehensive reasoning. Against this backdrop, we require an estimate of the exploration richness at each step. Drawing on prior work linking model's exploration space to semantic diversity (Cheng et al., 2025), we adopt semantic entropy (Kuhn et al., 2023) as a proxy. Specifically, we define a continuous switching signal $\sigma_t \in [0, 1]$ as the normalized entropy of the model's predicted continuation space:

$$\sigma_t = \frac{\mathcal{H}(s_t)}{\log_2 K}, \tag{4}$$

where $\mathcal{H}(s_t)$ denotes the entropy over K sampled next-token distributions at step t. This signal enables dynamic cognitive resource allocation pro-

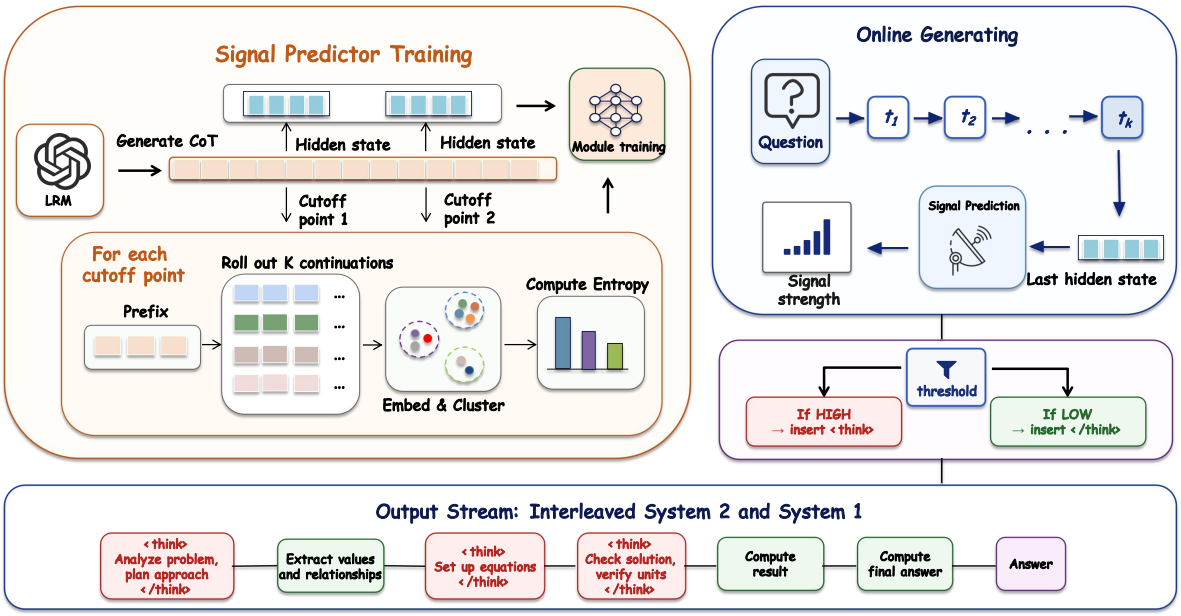

Figure 4: Overview of the InterSR framework. The signal prediction module extracts hidden states at sentence boundaries and predicts a system switching signal $\hat{\sigma}_t$. When $\hat{\sigma}_t$ exceeds the threshold τ, a control token is inserted to switch between System 1 and System 2 reasoning modes, enabling fine-grained cognitive resource allocation during inference.

portional to semantic uncertainty. As shown in Figure 3, the signal varies systematically across problem difficulty levels on DART-Math (Tong et al., 2024): harder problems (level 5) exhibit a higher proportion of High-signal steps, indicating greater System 2 demand during reasoning.

3.3 Capturing System Switching Signals

While the signal σ_t is constructed from semantic entropy, computing it requires extensive sampling and clustering, introducing substantial overhead. We investigate whether switching signals are already encoded in the model's internal states. As shown in Figure 2, hidden representations from the 7B and 14B models exhibit clear separation between high- and low-signal-strength points, indicating a strong correlation between signal strength and latent features. This suggests the switching signal can be effectively learned from internal activations. We introduce a lightweight module g_ϕ to predict signal strength from hidden states during reasoning.

3.4 Interleaved System-1/2 Reasoning

Based on the above findings, we propose an Interleaved System-1/2 Reasoning framework that enables language models to dynamically alternate between System 1 and System 2 thinking modes during generation for efficient and context-sensitive reasoning. The reasoning model treats <think> as the start of a slow-thinking segment and </think> as its end (Guo et al., 2025; Yang et al., 2025a), enabling explicit control over cognitive mode transitions within a single inference trajectory. At each generation step, the model first performs a forward pass to compute the hidden representation h_k based on the current context. To avoid disrupting output fluency, a lightweight boundary detector $f_\psi(h_k)$ identifies candidate sentence or reasoning-unit boundaries. Only at these boundaries does the model predict a switching signal $\hat{\sigma}_k$ from h_k using a learned predictor g_ϕ and compare it with a threshold τ. If the signal exceeds τ, a control token (<think> or </think>) is inserted to trigger a transition between System 1 and System 2. This ensures that mode switching occurs at semantically coherent regions, reducing token consumption while preserving reasoning performance. The full algorithm can be found in Appendix C.

4 Experiments

In this section, we thoroughly evaluate the performance of our proposed Interleaved System Reasoning framework by systematically addressing the following key research questions: **RQ1:** Can In-

Method	GSM8K		MATH500		AIME24		AMC23	
	Pass@1 (↑,%)	**Tokens (↓)**	**Pass@1 (↑,%)**	**Tokens (↓)**	**Pass@1 (↑,%)**	**Tokens (↓)**	**Pass@1 (↑,%)**	**Tokens (↓)**
DeepSeek-R1-Distill-Qwen-1.5B								
Vanilla	77.5	1265	84.8	4785	30.0	12496	72.5	7765
TALE	77.4 (-0.1)	1056 (-16%)	80.6 (-4.2)	3614 (-24%)	30.0 (+0.0)	11890 (-4%)	70.0 (-2.5)	7267 (-6%)
Dynasor	77.6 (+0.1)	1030 (-18%)	85.2 (+0.4)	3624 (-24%)	30.0 (+0.0)	11111 (-11%)	72.5 (+0.0)	6781 (-12%)
DEER	73.6 (−3.9)	820 (−35%)	80.6 (−4.2)	3770 (−21%)	26.7 (−3.3)	11845 (−5%)	67.5 (−5.0)	5958 (−23%)
Ours	**78.0 (+0.5)**	**614 (-51%)**	**86.0 (+1.2)**	**3504 (-26%)**	**33.3 (+3.3)**	**8061 (-35%)**	**72.5 (+0.0)**	**5922 (-23%)**
DeepSeek-R1-Distill-Qwen-7B								
Vanilla	86.4	743	88.6	3824	46.7	10953	90.0	5699
TALE	84.3 (-2.1)	633 (-14%)	88.0 (-0.6)	2941 (-23%)	50.0 (+3.3)	9640 (-11%)	87.5 (-2.5)	4649 (-18%)
Dynasor	86.5 (+0.1)	715 (-3%)	88.2 (-0.4)	2931 (-23%)	50.0 (+3.3)	9532 (-12%)	90.0 (+0.0)	4910 (-13%)
DEER	82.9 (−3.5)	712 (−4%)	84.4 (−4.2)	3020 (−21%)	43.3 (−3.4)	9871 (−10%)	85.0 (−5.0)	5546 (−3%)
Ours	**86.7 (+0.3)**	**517 (-30%)**	**89.6 (+1.0)**	**2651 (-30%)**	**50.0 (+3.3)**	**7259 (-33%)**	**90.0 (+0.0)**	**4126 (-27%)**
DeepSeek-R1-Distill-Llama-8B								
Vanilla	**84.7**	982	**70.8**	3677	36.7	10842	80.0	5518
TALE	83.2 (−1.5)	839 (−14%)	68.4 (−2.4)	2905 (−21%)	33.3 (−3.4)	9737 (−10%)	77.5 (−2.5)	4853 (−12%)
Dynasor	84.5 (−0.2)	938 (−4%)	70.6 (−0.2)	2868 (−22%)	36.7 (+0.0)	9573 (−12%)	80.0 (+0.0)	4855 (−12%)
DEER	81.7 (−3.0)	807 (−18%)	67.4 (−3.4)	3350 (−9%)	33.3 (−3.4)	10218 (−6%)	75.0 (−5.0)	5107 (−7%)
Ours	84.0 (−0.7)	**731 (−26%)**	70.2 (−0.6)	**2800 (−24%)**	**36.7 (+0.0)**	**7806 (−28%)**	**80.0 (+0.0)**	**4138 (−25%)**
DeepSeek-R1-Distill-Qwen-14B								
Vanilla	91.9	705	93.2	3591	63.3	9686	92.5	5435
TALE	91.1 (-0.8)	553 (-21%)	91.8 (-1.4)	2941 (-18%)	60.0 (-3.3)	**8450 (-12%)**	90.0 (-2.5)	4011 (-26%)
Dynasor	91.6 (-0.3)	663 (-5%)	93.0 (-0.2)	3317 (-7%)	63.3 (+0.0)	9054 (-6%)	92.5 (+0.0)	4023 (-25%)
DEER	88.5 (−3.4)	610 (−13%)	89.2 (−4.0)	2850 (−21%)	60.0 (−3.3)	8871 (−8%)	87.5 (−5.0)	4280 (−21%)
Ours	**92.0 (+0.1)**	**544 (-22%)**	**93.2 (+0.0)**	**2494 (-30%)**	**63.3 (+0.0)**	8557 (-11%)	**92.5 (+0.0)**	**3894 (-28%)**

Table 1: Performance comparison across reasoning strategies and datasets.

Metric	Vanilla	InterSR	Change
Avg. tokens/request	3824	2651	−30.7%
Peak KV memory (GB)	12.4	8.7	−29.8%
Throughput (req/s)	4.2	5.8	+38.1%
TTFT (ms)	142	145	+2.1%

Table 2: Serving efficiency on MATH500 (7B model, batch 32, 4×H200). InterSR's token reduction translates to substantial throughput gains.

terSR reduce the number of tokens generated by the model without sacrificing accuracy? **RQ2:** How does InterSR perform on datasets outside the mathematical reasoning domain? **RQ3:** Which hidden layer best captures the switching signal for prediction? **RQ4:** How do different thresholds for the system switching signal affect the model's performance and efficiency? **RQ5:** How can InterSR's ability to switch between systems for efficient reasoning be visually demonstrated and interpreted?

4.1 Experimental Settings

Benchmarks. We evaluate on four mathematical reasoning benchmarks spanning a wide range of difficulty: GSM8K (Cobbe et al., 2021) (grade-school arithmetic), MATH500 (Lightman et al., 2023) (competition-level problems), AMC23 (AMC competition), and AIME24 (AIME competition). These benchmarks require varied reasoning strategies from simple arithmetic to multi-step algebraic and geometric inference, enabling comprehensive evaluation of InterSR across difficulty levels.

Baselines. We compare against three representative adaptive reasoning strategies. TALE (Han et al., 2025) allocates a token budget based on problem complexity; we use the training-free TALE-EP variant for fair comparison. Dynasor (Fu et al., 2025) performs confidence-based early exit via Certaindex. DEER (Yang et al., 2025b) uses answer triggers to halt generation once the model reaches high confidence.

Experimental Models. We evaluate InterSR on four model scales: DeepSeek-R1-Distill-Qwen-1.5B/7B/14B and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025). The signal predictor g_ϕ and boundary detector f_ψ are lightweight MLPs operating on the final hidden state h_t, trained on ∼10k samples from DART-Math (Tong et al., 2024). Due to space constraints, full details on data construction, training procedures, and hyperparameters are provided in Appendix A.

4.2 Main Result (RQ1)

As shown in Table 1, our approach consistently reduces token usage across all datasets without sacrificing performance. On GSM8K, InterSR

Domain	Dataset	Model	Accuracy (%)			Tokens		
			Vanilla	InterSR	Diff	Vanilla	InterSR	Reduction
STEM	GPQA-D	R1-1.5B	26.8	28.3	+1.5	8777	4732	−46%
		R1-7B	46.5	47.5	+1.0	6751	3891	−42%
		R1-14B	50.5	51.0	+0.5	5899	3536	−40%
	ARC-C	R1-1.5B	57.9	57.8	−0.1	1164	1013	−13%
		R1-7B	84.1	83.6	−0.5	1025	789	−23%
		R1-14B	93.5	92.6	−0.9	705	515	−27%
Knowledge	MMLU	R1-1.5B	45.8	49.0	+3.2	1432	1274	−11%
		R1-7B	84.4	83.6	−0.8	1181	892	−24%
		R1-14B	84.4	83.8	−0.6	1170	772	−34%
Code	HumanEval	R1-1.5B	29.9	30.5	+0.6	1851	1420	−23%
		R1-7B	43.9	45.7	+1.8	1807	1128	−37%
		R1-14B	49.4	51.8	+2.4	1774	1087	−39%

Table 3: Cross-domain evaluation. **Diff**: InterSR − Vanilla accuracy change. **Reduction**: token reduction ratio.

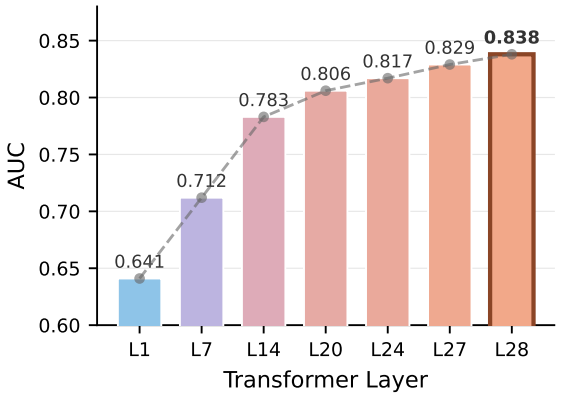

Figure 5: AUC of signal predictors trained on different hidden layers using the 7B model. Performance increases monotonically with depth.

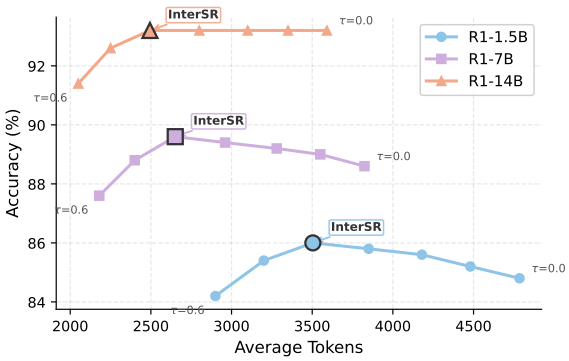

Figure 6: Efficiency versus accuracy Pareto front on MATH500. Each curve traces one model across threshold values $\tau = 0.0$ to 0.6.

achieves the largest savings—up to **51%** token reduction for the 1.5B model—since simpler problems contain more redundant reasoning steps that can be efficiently handled by System 1. On harder benchmarks like AIME24, savings are more modest (**26%** average), reflecting the genuine need for deeper reasoning on competition-level problems. Across all four model scales, InterSR either preserves or slightly improves accuracy: the 7B model gains +1.0% on MATH500 while saving 30% tokens, suggesting that suppressing redundant reasoning can actually prevent error accumulation. Compared to TALE, Dynasor, and DEER, InterSR achieves the best efficiency–accuracy trade-off: DEER aggressively truncates reasoning and suffers up to −5.0% accuracy loss on AMC23, while TALE and Dynasor offer limited token savings on difficult benchmarks. InterSR also yields practical serving gains: on MATH500 with the 7B model, throughput improves by 38% with 30% less KV cache memory (Table 2), demonstrating that token reduction directly translates to real-world deployment benefits. Details on computational overhead are in Appendix B.

4.3 Cross-domain Experiments (RQ2)

To investigate generalizability beyond mathematical reasoning, we evaluate InterSR on four diverse benchmarks: GPQA-Diamond (Rein et al., 2024) (graduate-level STEM), ARC-Challenge (Clark et al., 2018) (science reasoning), MMLU (Hendrycks et al., 2021) (knowledge-intensive QA), and HumanEval (Chen et al., 2021) (code generation). As shown in Table 3, InterSR achieves consistent token reduction (11%–46%) across all benchmarks and model sizes while maintaining comparable or improved accuracy. The largest savings appear on GPQA-Diamond, where the 1.5B model reduces tokens by 46% with a +1.5% accuracy gain, indicating that graduate-level STEM reasoning contains substantial redundancy that InterSR

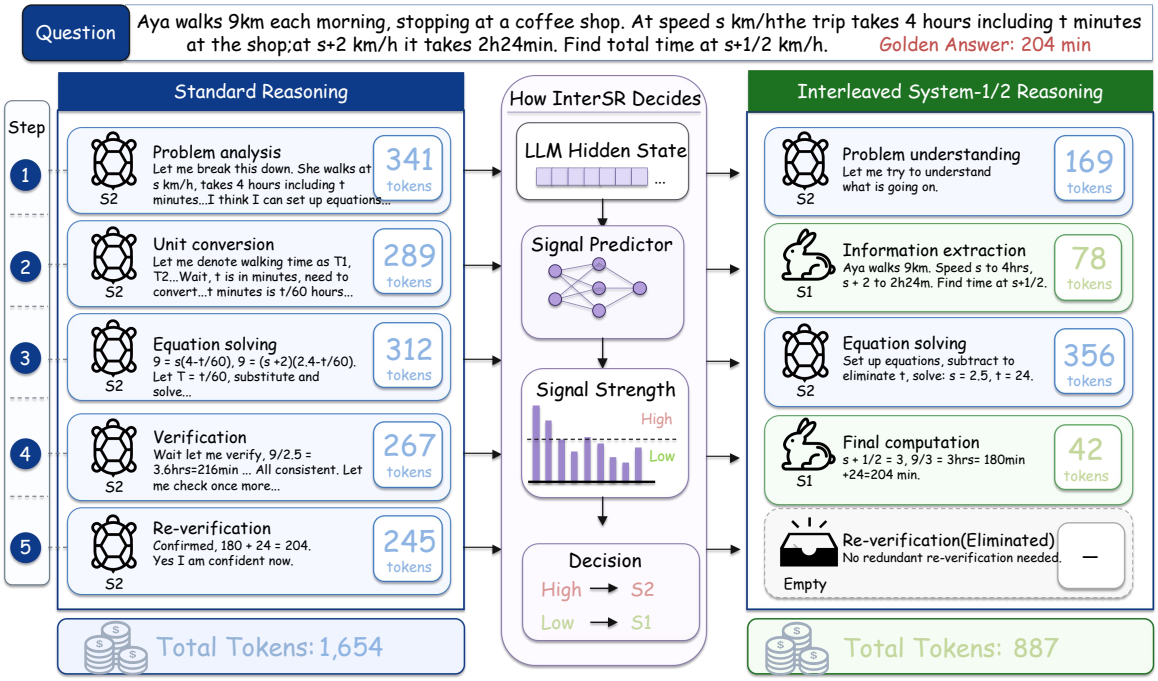

Figure 7: Case study: comparison of reasoning trajectories between Standard Reasoning and InterSR on an AIME24 problem. InterSR dynamically switches between System 1 and System 2 to minimize token consumption while maintaining reasoning quality.

effectively eliminates. Notably, on HumanEval, InterSR *improves* accuracy across all model sizes while reducing tokens by 23%–39%, suggesting that suppressing overthinking can benefit code generation by avoiding unnecessary re-derivation that introduces bugs. On knowledge-intensive tasks (MMLU), InterSR maintains accuracy within 1% while achieving 11%–34% token reduction. These results confirm that InterSR's switching signals, though trained on mathematical data, generalize effectively across diverse task domains.

4.4 Hidden Layer Selection for Signal Prediction (RQ3)

The signal predictor g_ϕ operates on hidden states from a specific transformer layer. To identify the optimal layer, we use the 7B model (28 layers) and train separate predictors on layers 1, 7, 14, 20, 24, 27, and 28 with identical architectures.

As shown in Figure 5, AUC increases monotonically from layer 1 (0.641) to the final layer 28 (0.838). Early layers yield weak predictions, as shallow representations primarily encode local syntactic features insufficient for estimating reasoning complexity. The final layers achieve the strongest performance, confirming that deeper representations encode richer uncertainty information relevant to system switching. We therefore use the final layer for signal prediction across all model sizes. Additional ROC analysis is in Appendix E.

4.5 Efficiency–Accuracy Trade-off (RQ4)

We sweep the switching threshold τ from 0.0 (vanilla) to 0.6 and plot accuracy against token count for each model on MATH500 (Figure 6). For all models, there exists a "sweet spot" range of thresholds where tokens decrease substantially while accuracy is preserved or improved—the 14B model maintains 93.2% accuracy from $\tau = 0.0$ to $\tau = 0.4$ while reducing tokens by 30.5%. Notably, moderate thresholds can *improve* accuracy: the 7B model achieves 89.6% at $\tau = 0.4$ vs. 88.6% at $\tau = 0.0$, suggesting that suppressing overthinking prevents error accumulation in intermediate reasoning steps. Beyond the sweet spot, aggressive thresholds ($\tau \geq 0.5$) begin to degrade accuracy as essential System 2 reasoning is suppressed. Larger models exhibit flatter curves, indicating greater robustness to aggressive thresholds—likely because their stronger representations allow System 1 to handle a wider range of reasoning steps effectively.

4.6 Case Study: InterSR vs. Standard (RQ5)

Figure 7 compares reasoning trajectories on an AIME24 problem. The standard process applies uniform System 2 across all five steps, consuming

1654 tokens with extensive redundant verification. In contrast, InterSR dynamically allocates cognitive resources: it begins in System 2 for problem understanding (169 tokens), then switches to System 1 for straightforward information extraction (78 tokens), re-engages System 2 for the core equation solving that requires careful algebraic manipulation (356 tokens), and returns to System 1 for simple final computation (42 tokens). Notably, the redundant re-verification step is entirely eliminated. This interleaved strategy requires only 887 tokens—a 46% reduction—by reserving deep reasoning for steps that genuinely demand it while compressing routine operations via System 1. Full case details are in Appendix N.

5 Related Work

In this section, we review previous methods developed to mitigate the overthinking problem in large reasoning models. These methods can be categorized into two distinct perspectives: Response Perspective and Inference Perspective.

Response Perspective. In this perspective, methods focus on reducing token usage by directly manipulating the reasoning output. Dynasor (Fu et al., 2025) introduces Certaindex, which helps the model exit reasoning early when no further tokens are likely to change the final answer. ConciseRL (Dumitru et al., 2025) guides the reinforcement learning process through a conciseness score, directing the model to generate more concise responses. DEER (Yang et al., 2025b) uses answer triggers to stop reasoning once the model reaches high confidence. ThinkPrune (Hou et al., 2025) encourages shorter responses by truncating reward lengths during reinforcement learning. VPG-EA (Chen et al., 2026b) uses variational posterior guidance to suppress redundant reasoning tokens while preserving logical verification steps. FCS (Chen et al., 2025c) fine-tunes models on efficient reasoning datasets with First-Correct Solutions to promote concise outputs. ES-CoT explores early stopping in chain-of-thought reasoning to terminate reasoning when answers converge (Mao et al., 2025), and RCP mining reasoning patterns to identify points for early exit (Wei et al., 2025). RPDM (Guan et al., 2026) monitors reasoning path deviation via high-entropy transition tokens to detect and terminate overthinking trajectories. These methods efficiently reduce token consumption by minimizing response length. However, they fail to account for the inherent complexity of the problem, which may result in overly condensed reasoning and a subsequent decline in performance.

Inference Perspective. Methods in this perspective focus on classifying problems by difficulty and dynamically selecting reasoning modes based on complexity, thereby minimizing unnecessary resource consumption. CKT-MHA (Qi et al., 2024) proposes an external policy to detect hard problems and adapt the reasoning strategy accordingly. FReM (Zhao et al., 2025) estimates the depth of reasoning and select prompts based on problem difficulty. TALE (Han et al., 2025) proposes “Token Budget" that limits reasoning to the minimum number of tokens needed to solve a problem. DynaThink (Pan et al., 2024) adjusts reasoning depth by sampling multiple responses to determine if a problem needs fast or slow thinking. AdaptThink (Zhang et al., 2025) introduces a balanced training regime to toggle between “thinking" and “no-thinking" modes. DiffAdapt (Liu et al., 2026) trains a lightweight probe on hidden states to classify problem difficulty and dynamically select per-question inference strategies. ReBalance (Li et al., 2026) uses confidence as a continuous indicator to steer reasoning trajectories, pruning redundancy during overthinking and promoting exploration during underthinking. While these methods account for problem difficulty, they often simplify it by categorizing problems as either easy or hard, neglecting the subtle semantic variations and the diverse reasoning steps involved in different problems.

6 Conclusion

In this work, we propose the Interleaved System-1/2 Reasoning (InterSR) framework, a novel approach to mitigate overthinking in large reasoning models by dynamically switching between System 1 and System 2 modes based on system signals. InterSR efficiently allocates reasoning resources, reducing token consumption while maintaining task performance. Extensive experiments across diverse datasets show that InterSR achieves substantial token savings, especially in simpler tasks, without compromising accuracy or reasoning quality. Cross-domain evaluations on GPQA-Diamond, MMLU, ARC-Challenge, and HumanEval further demonstrate consistent token reduction across diverse task domains. Our results demonstrate InterSR’s potential to enhance reasoning efficiency in LLMs and optimize performance across domains.

Limitations

While InterSR demonstrates substantial efficiency gains across multiple benchmarks, two directions remain for future exploration. First, although we evaluate on both Qwen-based and Llama-based architectures within the DeepSeek-R1-Distill family, extending InterSR to a wider range of model families (such as Mixtral or Gemma) would further validate its architectural generality. Second, our evaluation focuses on single-turn reasoning tasks; the effectiveness of InterSR in multi-turn agentic scenarios, where reasoning spans across multiple interaction rounds and requires long-horizon planning, remains an open question.

Ethical Considerations

This work focuses on improving the inference efficiency of large reasoning models without introducing new training data or user data collection, and thus does not raise additional privacy concerns beyond those of the underlying models. Nevertheless, reducing the cost of advanced reasoning may facilitate broader deployment, which could amplify both beneficial and harmful uses. We encourage the use of InterSR with appropriate safeguards, particularly in high-stakes settings where reasoning errors may cause real-world harm. In addition, because dynamic switching may shorten reasoning traces, future work should further examine its impact on transparency, interpretability, and human oversight.

References

Marthe Ballon, Andres Algaba, and Vincent Ginis. 2025. The relationship between reasoning and performance in large language models–o3 (mini) thinks harder, not longer. *arXiv preprint arXiv:2502.15631*.

Yi Bin, Tianyi Jiang, Yujuan Ding, Kainian Zhu, Fei Ma, Jingkuan Song, Yang Yang, and Heng Tao Shen. 2026. Explore briefly, then decide: Mitigating llm overthinking via cumulative entropy regulation. *arXiv preprint arXiv:2510.02249*.

Shelly Chaiken and Yaacov Trope. 1999. *Dual-process theories in social psychology*. Guilford Press.

Hanting Chen, Yasheng Wang, Kai Han, Dong Li, Lin Li, Zhenni Bi, Jinpeng Li, Haoyu Wang, Fei Mi, Mingjian Zhu, and 1 others. 2025a. Pangu embedded: An efficient dual-system llm reasoner with metacognition. *arXiv preprint arXiv:2505.22375*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wei Chen, Han Ding, Meng Yuan, Zhao Zhang, Deqing Wang, and Fuzhen Zhuang. 2025b. Bridging social psychology and llm reasoning: Conflict-aware meta-review generation via cognitive alignment. *arXiv preprint arXiv:2503.13879*.

Wei-Lin Chen, Liqian Peng, Tian Tan, Chao Zhao, Blake JianHang Chen, Ziqian Lin, Alec Go, and Yu Meng. 2026a. Think deep, not just long: Measuring llm reasoning effort via deep-thinking tokens. *arXiv preprint arXiv:2602.13517*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2025c. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. In *Proceedings of the 42nd International Conference on Machine Learning*.

Zizhao Chen, Yuying Li, Siting Lin, and Lianxi Wang. 2026b. Efficient llm reasoning via variational posterior guidance with efficiency awareness. *arXiv preprint arXiv:2605.11019*.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Julian Coda-Forno, Zhuokai Zhao, Qiang Zhang, Dipesh Tamboli, Weiwei Li, Xiangjun Fan, Lizhu Zhang, Eric Schulz, and Hsiao-Ping Tseng. 2025. Exploring system 1 and 2 communication for latent reasoning in llms. *arXiv preprint arXiv:2510.00494*.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Zheng Du, Hao Kang, Song Han, Tushar Krishna, and Ligeng Zhu. 2026. Ockbench: Measuring the efficiency of llm reasoning. *arXiv preprint arXiv:2511.05722*.

Razvan-Gabriel Dumitru, Darius Peteleaza, Vikas Yadav, and Liangming Pan. 2025. Conciserl: Conciseness-guided reinforcement learning for efficient reasoning models. *arXiv preprint arXiv:2505.17250*.

Jonathan St BT Evans. 2017. Dual process theory: Perspectives and problems. *Dual process theory 2.0*, pages 137–155.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and 1 others. 2025. Efficiently scaling llm reasoning with certaindex. In *Advances in Neural Information Processing Systems*.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. 2025. Does thinking more always help? understanding test-time scaling in reasoning models. *arXiv preprint arXiv:2506.04210*.

Weixin Guan, Liang Li, Jiapeng Liu, Bing Li, Peng Fu, Chengyang Fang, Xiaoshuai Hao, Can Ma, and Weiping Wang. 2026. Mitigating overthinking in large reasoning language models via reasoning path deviation monitoring. *arXiv preprint arXiv:2603.14251*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Nature*, 645:633–638.

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.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Yulin Li, Tengyao Tu, Li Ding, Junjie Wang, Huiling Zhen, Yixin Chen, Yong Li, and Zhuotao Tian. 2026. Efficient reasoning with balanced thinking. *arXiv preprint arXiv:2603.12372*.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. 2026. Diffadapt: Difficulty-adaptive reasoning for token-efficient llm inference. *arXiv preprint arXiv:2510.19669*.

Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. 2025. Early stopping chain-of-thoughts in large language models. *arXiv preprint arXiv:2509.14004*.

Jiabao Pan, Yan Zhang, Chen Zhang, Zuozhu Liu, Hongwei Wang, and Haizhou Li. 2024. Dynathink: Fast or slow? a dynamic decision-making framework for large language models. *arXiv preprint arXiv:2407.01009*.

Biqing Qi, Xinquan Chen, Junqi Gao, Dong Li, Jianxing Liu, Ligang Wu, and Bowen Zhou. 2024. Interactive continual learning: Fast and slow thinking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12892.

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*.

Swarnadeep Saha, Archiki Prasad, Justin Chih-Yao Chen, Peter Hase, Elias Stengel-Eskin, and Mohit Bansal. 2024. System-1. x: Learning to balance fast and slow planning with language models. *arXiv preprint arXiv:2407.14414*.

Jihao Shi, Xiao Ding, Hengwei Zhao, Ting Liu, and Bing Qin. 2025. Bridging neural and symbolic reasoning: A dual-system framework for interpretable question answering. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Elizabeth P Shulman, Ashley R Smith, Karol Silva, Grace Icenogle, Natasha Duell, Jason Chein, and Laurence Steinberg. 2016. The dual systems model: Review, reappraisal, and reaffirmation. *Developmental cognitive neuroscience*, 17:103–117.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. 2025. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *Advances in Neural Information Processing Systems*, 37:7821–7846.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zihao Wei, Liang Pang, Jiahao Liu, Jingcheng Deng, Shicheng Xu, Zenghao Duan, Jingang Wang, Fei Sun, Xunliang Cai, Huawei Shen, and Xueqi Cheng. 2025. Stop spinning wheels: Mitigating llm overthinking via mining patterns for early reasoning exit. *arXiv preprint arXiv:2508.17627*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv 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. *arXiv preprint arXiv:2504.15895*.

Wenlin Yao, Haitao Mi, and Dong Yu. 2024. Hdflow: Enhancing llm complex problem-solving with hybrid thinking and dynamic workflows. *arXiv preprint arXiv:2409.17433*.

Zhifeng Yu, Yang Liu, and Xiaodong Wang. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025. Adaptthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*.

Zhengyi Zhao, Shubo Zhang, Zezhong Wang, Bin Liang, Binyang Li, and Kam-Fai Wong. 2025. Frem: A flexible reasoning mechanism for balancing quick and slow thinking in long-context question answering. *arXiv preprint arXiv:2503.22985*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kober, Yineng Liang, Siyuan Zhuang, and 1 others. 2024. Sglang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*.

Shu Zhou, Rui Ling, Junan Chen, Xin Wang, Tao Fan, and Hao Wang. 2026. When more thinking hurts: Overthinking in llm test-time compute scaling. *arXiv preprint arXiv:2604.10739*.

Alireza S Ziabari, Nona Ghazizadeh, Zhivar Sourati, Farzan Karimi-Malekabadi, Payam Piray, and Morteza Dehghani. 2025. Reasoning on a spectrum: Aligning llms to system 1 and system 2 thinking. *arXiv preprint arXiv:2502.12470*.

A Experiment Details

This appendix provides a comprehensive description of the experimental setup used in this paper, including model architectures, auxiliary modules, training data construction, optimization details, inference configurations, evaluation protocols, and hyperparameter settings, to facilitate reproducibility.

A.1 Models and Auxiliary Modules

A.1.1 Base Models

We evaluate our proposed InterSR framework on three model scales from the DeepSeek-R1-Distill-Qwen family (1.5B, 7B, 14B) and DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025). These models are distilled versions of the Qwen-2.5 and Llama-3.1 architectures, optimized for reasoning-centric tasks. The corresponding hidden dimensions are 1536, 3584, 5120, and 4096, respectively.

A.1.2 Signal Prediction Modules

The InterSR framework introduces two lightweight auxiliary modules for adaptive reasoning control.

The boundary detector f_ψ is implemented as a two-layer MLP with ReLU activation and a dropout

rate of 0.1. The signal predictor g_ϕ is a three-layer MLP with ReLU activations, a dropout rate of 0.1, and a final Sigmoid layer that outputs normalized signal values $\sigma_t \in [0, 1]$.

Both modules operate in bfloat16 precision to improve computational efficiency.

A.2 Training Data Construction

A.2.1 Dataset and Problem Sampling

To train the signal prediction modules, we construct labeled data using the DART-Math dataset (Tong et al., 2024). Specifically, for each difficulty level (1–5), we sample 1,000 problems and generate full Chain-of-Thought solutions using the corresponding base model.

Due to the large scale of the original dataset, this sampling strategy ensures a balanced and manageable training set across difficulty levels.

A.2.2 Reasoning Trace Segmentation

The generated reasoning traces are segmented into sentence-level units using the Stanza syntactic analysis toolkit. Each sentence boundary is treated as a candidate decision point for system switching.

A.2.3 Semantic Entropy Estimation

At each sentence boundary, we sample $K = 8$ candidate continuations from the model and compute their semantic embeddings using the Qwen3-Embedding-0.6B model (Yang et al., 2025a).

Semantic entropy is computed via incremental clustering with a distance threshold $\delta = 0.5$, and normalized by $\log_2 K$. The resulting entropy values serve as supervision signals for training the signal predictor.

For each model size, we construct approximately 10k training samples, with a 90%/10% train/test split.

A.3 Module Training

We train both auxiliary modules using the AdamW optimizer with a learning rate of 2×10^{-4}. The batch size is set to 128 for both modules.

For the boundary detector f_ψ, we employ Focal Loss with $\alpha = 1$ and $\gamma = 2$ to address class imbalance between boundary and non-boundary positions.

For the signal predictor g_ϕ, we use a weighted mean squared error (MSE) loss, assigning a weight of 0.1 to zero-valued labels.

Training is conducted with early stopping based on validation performance, with a patience of 3–5 evaluation rounds. Validation is performed every 10 batches throughout training.

A.4 Inference Configuration

During inference, we set the maximum generation length to 16,384 tokens, the temperature to 0.6, and top-p to 0.95 with nucleus sampling enabled.

The system switching threshold τ is tunable. In our main experiments, we explore values ranging from 0.0 to 0.6 to analyze the efficiency–accuracy trade-off.

To ensure stable mode transitions, we enforce:

- A minimum of 10 tokens in System 2 mode before allowing exit, and
- At least 10 tokens in System 1 mode before re-entering System 2.

A.5 Evaluation Setup

All experiments are conducted on 4 $\times$ NVIDIA H200 GPUs.

We evaluate InterSR on four mathematical reasoning benchmarks—GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), AIME24, and AMC23—as well as four cross-domain benchmarks: GPQA-Diamond (Rein et al., 2024) (graduate-level STEM), MMLU (knowledge-intensive QA), ARC-Challenge (science reasoning), and HumanEval (code generation).

We report Pass@1 accuracy and average token count as primary evaluation metrics. Answer correctness is verified using the `math-verify` toolkit for mathematical expression parsing and equivalence checking.

A.6 Baselines

We compare InterSR against two state-of-the-art adaptive reasoning methods:

- TALE (Han et al., 2025), which controls generation length via token budget allocation based on problem complexity. We use the training-free TALE-EP variant for fair comparison.
- Dynasor (Fu et al., 2025), which employs confidence-based early exit using Certaindex. We modify the original implementation to ensure complete solution output upon early termination.
- DEER (Yang et al., 2025b), a training-free early-exit method that uses answer triggers to halt generation early.

A.7 Datasets

Below we describe the datasets used in our experiments. Due to differences in HuggingFace data formats, the field names in `evaluation_config.yaml` must be adjusted accordingly.

AIME24 is derived from the 2024 American Invitational Mathematics Examination and is available on HuggingFace.[2] In our evaluation, the "question" field corresponds to the "problem" field, and the "answer" field maps to the "solution" field.

AMC23 is based on the 2023 American Mathematics Competition.[3] The "question" field corresponds to the dataset's "question" field, and the "answer" field maps to the "answer" field.

MATH500 is a curated subset of the MATH benchmark.[4] The "question" field corresponds to "problem", and the "answer" field corresponds to "answer".

GSM8K is a grade school math word problem benchmark.[5] The "question" and "answer" fields are directly used.

GPQA-Diamond is the highest-difficulty subset of GPQA.[6] We use the "diamond" split and convert the original format into multiple-choice questions by randomly shuffling the provided options and assigning them to options A–D.

MMLU (Massive Multitask Language Understanding) is a comprehensive knowledge benchmark covering 57 subjects.[7] We evaluate on the full test set and report overall accuracy.

ARC-Challenge is the challenge subset of the AI2 Reasoning Challenge, containing science questions that require complex reasoning.[8] We use the "ARC-Challenge" split.

HumanEval is a code generation benchmark consisting of 164 hand-crafted programming problems.[9] We report Pass@1 accuracy using execution-based evaluation.

DART-Math, We use the **Dart Math Pool Math** subset of DART-Math.[10] We sample 1,000 problems from each difficulty level to construct a balanced training set for signal prediction and boundary detection.

Models	Datasets			
	AIME24	AMC23	MATH500	GSM8K
R1-1.5B	0.5	0.6	0.4	0.6
R1-7B	0.2	0.4	0.4	0.4
R1-8B	0.2	0.4	0.4	0.4
R1-14B	0.2	0.3	0.4	0.3

Table 4: Signal thresholds used by each model across datasets.

A.8 Hyperparameter Settings

A.8.1 Semantic Entropy Calculation

Semantic entropy is computed via clustering of sampled response embeddings. We set the distance threshold to $\delta = 0.5$. Two responses with embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ are merged into the same cluster if:

$$\|\mathbf{e}_i - \mathbf{e}_j\| < \delta.$$

The centroid of the merged cluster is computed as:

$$\mathbf{e}_C = \frac{1}{|C_i| + |C_j|} \sum_{k \in C_i \cup C_j} \mathbf{e}_k.$$

After clustering, suppose there are K clusters with frequencies p_k. The semantic entropy $\mathcal{H}$ is defined as:

$$\mathcal{H} = -\sum_{k=1}^{K} p_k \log p_k.$$

A.8.2 System Switching Thresholds

For smaller models, we adopt higher thresholds since their intrinsic signal strengths tend to be larger. Conversely, larger models require lower thresholds to trigger System 2 appropriately.

For the particularly challenging AIME24 dataset, we use lower thresholds to encourage more frequent engagement of System 2, ensuring sufficient deliberation and maintaining strong performance.

[2] https://huggingface.co/datasets/math-ai/aime24
[3] https://huggingface.co/datasets/math-ai/amc23
[4] https://huggingface.co/datasets/HuggingFaceH4/MATH-500
[5] https://huggingface.co/datasets/openai/gsm8k
[6] https://huggingface.co/datasets/Idavidrein/gpqa
[7] https://huggingface.co/datasets/cais/mmlu
[8] https://huggingface.co/datasets/allenai/ai2_arc
[9] https://huggingface.co/datasets/openai/openai_humaneval
[10] https://huggingface.co/datasets/hkust-nlp/dart-math-pool-math

B Cost Analysis

In this section, we analyze the computational overhead introduced by InterSR's signal prediction module from two perspectives: parameter count and inference speed.

B.1 Parameter Analysis

The signal prediction module in InterSR consists of a boundary detector f_ψ (two-layer MLP) and a signal predictor g_ϕ (three-layer MLP). Table 5 reports the parameter counts for each model scale.

Model	Base	InterSR	Ratio
R1-1.5B	1.5B	4.7M	0.31%
R1-7B	7.6B	25.7M	0.34%
R1-8B	8.0B	33.6M	0.42%
R1-14B	14.7B	52.4M	0.36%

Table 5: Parameter overhead of the InterSR signal prediction module across model scales.

Across all model scales, the signal prediction module adds at most 52.4M parameters, accounting for no more than 0.42% of the base model size. This lightweight design ensures that InterSR can be integrated into existing models with negligible memory overhead, making it practical for both research and production deployment.

B.2 Token Efficiency Analysis

We further investigate the model's generation efficiency on a per-token basis, as shown in Figure 8. The results indicate that our method increases the per-token generation time by less than 3%, while simultaneously achieving an average reduction of over 30% in overall token usage compared to the standard InterSR approach. This minimal per-token overhead, combined with the significant savings in token consumption, highlights the efficiency of our Interleaved System-1/2 Reasoning framework. By dynamically allocating cognitive resources and avoiding unnecessary overthinking, the method not only reduces redundant computations but also accelerates the path to the correct solution. These findings demonstrate that our approach effectively balances computational cost and reasoning quality, ultimately enhancing the overall efficiency and practicality of large language model reasoning.

C Algorithm

InterSR generates tokens by dynamically switching between fast (System 1) and slow (System 2) reasoning. At each step, a learned trigger decides whether to switch, inserting special tokens for slow thinking when needed. Otherwise, tokens are generated normally. This allows the model to focus slow, careful reasoning on critical steps while handling simpler parts quickly, improving overall efficiency. The full algorithm is illustrated in Algorithm 1.

Algorithm 1 Interleaved Reasoning with System Switching

Require: Prompt x, reasoning model M, switch trigger f_ψ, confidence predictor g_ϕ, threshold τ, initial mode $m \in \{\mathrm{S}_1, \mathrm{S}_2\}$, maximum length L_{max}

Ensure: Token sequence t

```
t ← [] {Initialize output token sequence}
k ← 0 {Initialize step counter}
while k ≤ L_max do
    k ← k + 1 {Increment step counter}
    h_k ← M(x, t) {Compute current hidden state from model}
    if f_ψ(h_k) ≥ 0 then
        σ̂_k ← g_φ(h_k) {Compute switching signal strength}
        if σ̂_k ≥ τ and m = S_1 then
            t ← t ∪ {<think>} {Insert System 2 thinking token}
            m ← S_2 {Switch to System 2}
            continue {Skip current token generation}
        else if σ̂_k < τ and m = S_2 then
            t ← t ∪ {</think>} {Insert System 1 return token}
            m ← S_1 {Switch to System 1}
            continue {Skip current token generation}
        end if
    end if
    t_k ~ softmax(W h_k) {Generate token normally from hidden state}
    t ← t ∪ {t_k} {Append generated token to output sequence}
    if t_k is end-of-sequence then
        break {Stop generation at EOS token}
    end if
end while
return t {Return final token sequence}
```

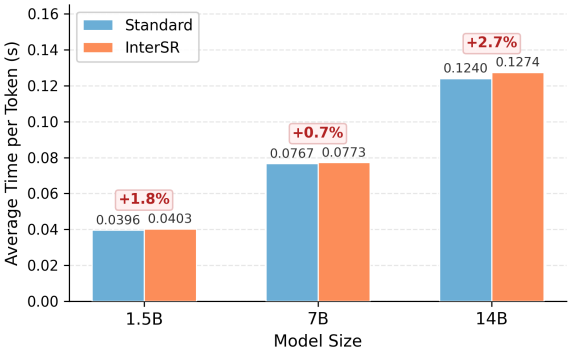

Figure 8: Comparison of token generation efficiency between InterSR and standard reasoning across different model sizes.

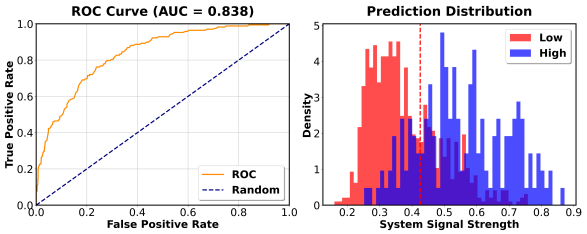

Figure 9: ROC curve and predicted score distribution for the system signal predictor of DeepSeek-R1-Distill-Qwen-7B.

D Control Tag Distribution Analysis

A natural concern is whether the `<think>` and `</think>` control tags inserted by InterSR genuinely induce different reasoning behaviors, or merely act as superficial markers with no impact on the generation process. To address this, we conduct a systematic analysis of the next-token distributions following each tag type.

Setup. We collect all reasoning traces generated by the DeepSeek-R1-Distill-Qwen-7B model under InterSR on the GSM8K and MATH500 datasets. For each occurrence of `<think>` (entering System 2) and `</think>` (exiting to System 1), we record the probability distribution over the vocabulary for the immediately following token. We then aggregate these distributions across all instances to obtain two average next-token distributions: P_{enter} (after `<think>`) and P_{exit} (after `</think>`).

Jensen-Shannon Divergence. We compute the Jensen-Shannon Divergence (JSD) between P_{enter} and P_{exit}:

$$\text{JSD}(P_{\text{enter}} \| P_{\text{exit}}) = \frac{1}{2} D_{\text{KL}}(P_{\text{enter}} \| M) + \frac{1}{2} D_{\text{KL}}(P_{\text{exit}} \| M) \quad (5)$$

where $M = \frac{1}{2}(P_{\text{enter}} + P_{\text{exit}})$. The resulting JSD is **0.83** (on a $[0, 1]$ scale), indicating a strong distributional shift between the two modes. As a reference, JSD values above 0.5 are generally considered indicative of substantially different distributions.

Top-Token Overlap. We further examine the top-50 most probable tokens under each distribution. The overlap between the two sets is only **8.2%** (approximately 4 shared tokens out of 50). After `<think>`, the model predominantly generates tokens associated with deliberative reasoning (such as mathematical operators, "let", "consider", "suppose"), while after `</think>`, the distribution favors direct, concise tokens (such as "so", "therefore", digits, punctuation).

These findings confirm that InterSR's mode switching induces genuine cognitive regime changes at the token level, rather than superficial tag insertion with no downstream effect.

E Signal Predictor ROC Analysis

We further evaluate whether the learned predictor can reliably capture the switching signal from hidden states. We binarize the continuous signal intensity by applying a threshold and evaluate the predictor's discrimination ability. As shown in Figure 9, the predictor achieves an AUC of 0.838, with limited overlap between predicted scores for high- and low-intensity signals, confirming that discriminative features can be effectively extracted from hidden states.

F SGLang Integration

To facilitate practical deployment of InterSR in production environments, we integrate it into the SGLang inference framework (Zheng et al., 2024), a high-performance serving engine for large language models. SGLang provides continuous batching, PagedAttention, and RadixAttention optimizations that are essential for efficient serving. Our integration preserves all these optimizations while enabling real-time InterSR-based reasoning mode control.

Integration Mechanism. SGLang supports the `SGLANG_EXTERNAL_MODEL_PACKAGE` environment variable, which allows users to register custom model wrappers that intercept the forward pass without modifying SGLang's core codebase.

We leverage this mechanism to inject InterSR's signal prediction logic.

Implementation Details. The integration consists of the following components:

1. **Custom Model Wrapper.** We define a model class that inherits from SGLang's base model interface and wraps the original DeepSeek model. The trained boundary detector f_ψ and signal predictor g_ϕ are attached as additional submodules and loaded alongside the base model weights.

2. **Hidden State Extraction.** During each forward pass, the wrapper hooks into the final transformer layer to extract the hidden state h_k. This is implemented via PyTorch forward hooks, adding negligible overhead.

3. **Signal Evaluation and Token Injection.** At each generation step, the boundary detector $f_\psi(h_k)$ first determines whether the current position is a candidate boundary. If so, the signal predictor $g_\phi(h_k)$ computes the switching signal $\hat{\sigma}_k$. When a mode transition is triggered (i.e., $\hat{\sigma}_k \geq \tau$ in System 1 or $\hat{\sigma}_k < \tau$ in System 2), the wrapper injects the appropriate control token (`<think>` or `</think>`) into the generation stream by modifying the next-token logits.

4. **State Management.** The current reasoning mode (System 1 or System 2) and the token count since the last mode switch are maintained per request in SGLang's request state, ensuring correct behavior under continuous batching where multiple requests are processed concurrently.

Logits Modification for Mode Switching. The core of the integration lies in how mode switching decisions are translated into the autoregressive generation process. Rather than externally interrupting the decoding loop, we embed the switching logic directly into the model's logits output via an indicator-gated logits modification mechanism.

Let $\ell_k \in \mathbb{R}^{|V|}$ denote the raw logits produced by the model at step k, where $|V|$ is the vocabulary size. We define a mode switching indicator function $\mathbb{I}_{\text{switch}}(k)$ as:

$$\mathbb{I}_{\text{switch}}(k) = \mathbb{I}[f_\psi(h_k) \geq 0] \cdot \begin{cases} \mathbb{I}[\hat{\sigma}_k \geq \tau], & \text{if } m_k = \text{S}_1 \\ \mathbb{I}[\hat{\sigma}_k < \tau], & \text{if } m_k = \text{S}_2 \end{cases} \tag{6}$$

where $f_\psi(h_k) \geq 0$ indicates a candidate boundary, $\hat{\sigma}_k = g_\phi(h_k)$ is the predicted signal strength, τ is the switching threshold, and $m_k \in \{\text{S}_1, \text{S}_2\}$ is the current reasoning mode.

When $\mathbb{I}_{\text{switch}}(k) = 1$, we modify the logits to force the generation of the appropriate control token. Let t_{ctrl} denote the target control token ID (`<think>` when switching $\text{S}_1 \to \text{S}_2$, or `</think>` when switching $\text{S}_2 \to \text{S}_1$). The modified logits $\tilde{\ell}_k$ are computed as:

$$\tilde{\ell}_k[v] = \begin{cases} \ell_k[v] + \lambda, & \text{if } v = t_{\text{ctrl}} \\ -\infty, & \text{otherwise} \end{cases} \tag{7}$$

where λ is a large positive constant (we use $\lambda = 10^4$) that ensures the control token is selected with probability approaching 1 after softmax normalization. Formally, the probability of generating the control token becomes:

$$P(t_{\text{ctrl}} \mid h_k) = \frac{\exp(\tilde{\ell}_k[t_{\text{ctrl}}])}{\sum_{v \in V} \exp(\tilde{\ell}_k[v])} \approx 1 \tag{8}$$

This logits-level injection approach has several advantages: (1) it operates entirely within the model's forward pass, requiring no modification to SGLang's sampling kernel or decoding loop; (2) it is compatible with batched inference, as the logits modification can be applied independently per request; (3) it preserves the KV cache structure, since the injected control token is processed as a regular token in subsequent steps.

Minimum Mode Duration Constraint. To prevent rapid oscillation between modes (which could degrade output coherence), we enforce a minimum mode duration constraint. Let n_k denote the number of tokens generated since the last mode switch. The effective indicator becomes:

$$\tilde{\mathbb{I}}_{\text{switch}}(k) = \mathbb{I}_{\text{switch}}(k) \cdot \mathbb{I}[n_k \geq n_{\min}] \tag{9}$$

where $n_{\min} = 10$ is the minimum number of tokens required in each mode before a switch is permitted. This constraint prevents degenerate behavior where the model rapidly alternates between modes at every boundary.

Deployment. Users can deploy InterSR-enhanced models by simply setting the environment variable and pointing to the custom model package:

```
SGLANG_EXTERNAL_MODEL_PACKAGE=\
  isr_model \
python -m sglang.launch_server \
  --model deepseek-r1-distill-qwen-7b
```

Performance. The integration adds less than 3% overhead to per-token generation time, consistent with our standalone evaluation results (see Appendix B). The overhead comes primarily from the MLP forward passes of f_ψ and g_ϕ, which together contain fewer than 52.4M parameters (see Table 5). The logits modification itself is a constant-time operation that adds negligible latency.

G Failure Case Analysis

While InterSR achieves consistent efficiency gains across benchmarks, it is important to understand its failure modes. We conduct a systematic analysis of cases where InterSR produces incorrect answers that the standard (vanilla) model answers correctly. We identify two primary failure patterns:

Premature System 1 Transition. The most common failure mode occurs when InterSR transitions from System 2 to System 1 too early during a critical reasoning step. This typically happens when the signal predictor underestimates the complexity of an upcoming derivation. Consider a multi-step algebraic problem where the model needs to expand a polynomial expression. If the signal drops below the threshold τ after the model has set up the equation but before completing the expansion, InterSR switches to System 1. In System 1 mode, the model tends to skip intermediate steps, which can lead to arithmetic errors or incorrect simplifications.

Formally, let r_t be a reasoning step where full System 2 reasoning is required (i.e., $\mathcal{G}(s_t, 0) < T$), but the predictor incorrectly estimates $\hat{\sigma}_t < \tau$. The resulting information loss can be quantified as:

$$\Delta\mathcal{G}_t = \mathcal{G}(s_t, 1) - \mathcal{G}(s_t, 0) > 0 \qquad (10)$$

When $\Delta\mathcal{G}_t$ is large (i.e., the step is genuinely complex), premature switching leads to critical information loss that propagates through subsequent reasoning steps. In our analysis of the 7B model on MATH500, approximately 62% of InterSR-specific errors (cases where InterSR fails but vanilla succeeds) involve at least one premature $S_2 \rightarrow S_1$ transition during a critical derivation step.

Code and Symbolic Syntax Disruption. The second failure pattern is specific to tasks involving structured syntax, such as mathematical notation or code generation. When InterSR inserts a `</think>` tag mid-expression, it can disrupt the syntactic structure of the output. For example, in a LaTeX-formatted mathematical derivation, a mode switch between `\frac{` and its closing brace can cause the model to lose track of the nesting structure, producing malformed expressions.

This issue is more prevalent in the 1.5B model, where the boundary detector f_ψ has lower precision due to the smaller hidden dimension (1536 vs. 3584 for 7B). In our analysis, syntax disruption accounts for approximately 23% of InterSR-specific errors on the 1.5B model but only 8% on the 14B model, suggesting that larger models develop more robust representations of syntactic boundaries.

Error Rate Analysis. We quantify the overall failure rate by examining the fraction of problems where InterSR answers incorrectly but vanilla answers correctly (InterSR-specific errors):

Model	GSM8K	MATH500	AIME24	AMC23
R1-1.5B	3.2%	4.6%	6.7%	5.0%
R1-7B	1.8%	2.4%	3.3%	2.5%
R1-8B	2.2%	2.8%	3.3%	2.5%
R1-14B	1.1%	1.4%	3.3%	2.5%

Table 6: InterSR-specific error rate (problems where InterSR fails but vanilla succeeds). These are offset by cases where InterSR succeeds but vanilla fails, resulting in the net accuracy differences shown in Table 1.

Importantly, these InterSR-specific errors are partially offset by cases where InterSR *improves* over vanilla (by avoiding overthinking-induced errors), resulting in the net accuracy changes reported in our main results. The decreasing error rate with model scale suggests that larger models produce more reliable signal predictions, enabling more accurate mode switching decisions.

H KV Cache Efficiency Analysis

A practical concern for deploying InterSR in production serving systems is whether the insertion of control tokens (`<think>` and `</think>`) and the resulting mode switches affect the KV cache structure and inference efficiency. We provide a detailed analysis below.

KV Cache Compatibility. The control tokens inserted by InterSR are standard tokens in the model's vocabulary. When a control token is injected via the logits modification mechanism (see Appendix F), it is processed by the model in the same way as any other generated token: the model computes its attention over all previous KV entries and appends its own key-value pair to the cache. Crucially, this means that:

1. The KV cache remains **contiguous and complete**—no entries are deleted, reordered, or invalidated during a mode switch.
2. The attention mechanism can attend to all previous tokens regardless of which reasoning mode produced them, preserving full contextual continuity.
3. No cache invalidation or recomputation is required, unlike methods that modify or rollback previously generated tokens.

Memory Savings. Since InterSR reduces the total number of generated tokens (by 11%–51% across benchmarks), it directly reduces the peak KV cache memory consumption. For a model with L layers, hidden dimension d, and n generated tokens, the KV cache memory is:

$$\mathcal{M}_{\text{KV}} = 2 \cdot L \cdot n \cdot d \cdot b \tag{11}$$

where b is the bytes per element (2 for bfloat16). Since InterSR reduces n by a factor of $(1 - \alpha)$ where α is the token reduction ratio, the memory savings are proportional:

$$\Delta\mathcal{M}_{\text{KV}} = 2 \cdot L \cdot \alpha \cdot n_{\text{vanilla}} \cdot d \cdot b \tag{12}$$

For the 7B model ($L = 28$, $d = 3584$) with an average 30% token reduction on MATH500 (from 3824 to 2651 tokens), this translates to:

$$\Delta\mathcal{M} = 2 \times 28 \times 1173 \times 3584 \times 2 \approx 472 \text{ MB} \tag{13}$$

per request. In a serving scenario with multiple concurrent requests, these savings are multiplicative, enabling higher throughput or larger batch sizes under the same memory budget. The detailed serving throughput comparison is presented in Table 2 in the main text.

I System 1/2 Mode Distribution

A key design principle of InterSR is that it adaptively allocates reasoning effort based on problem complexity. To verify this, we analyze the proportion of tokens generated under System 1 (fast) versus System 2 (slow) across benchmarks and model scales.

Setup. For each model (1.5B, 7B, 14B), we run InterSR on all four mathematical benchmarks and record the number of tokens generated in each reasoning mode. Tokens within `<think>...</think>` segments are counted as System 2 tokens; all other tokens are counted as System 1.

Results. As shown in Figure 10, the System 2 token ratio increases consistently with benchmark difficulty across all model scales. On GSM8K (easiest), the 7B model allocates only 28% of tokens to System 2, while on AIME24 (hardest), this ratio rises to 60%. This confirms that InterSR's signal-driven switching mechanism correctly identifies and responds to varying levels of problem complexity.

Two additional patterns emerge across model scales:

1. **Larger models use less System 2.** For the same benchmark, larger models consistently allocate a smaller proportion of tokens to System 2. On GSM8K, the S2 ratio decreases from 35% (1.5B) to 28% (7B) to 22% (14B). This reflects the superior reasoning capability of larger models—they can resolve more reasoning steps through fast System 1 processing without sacrificing accuracy.
2. **Difficulty sensitivity scales with model size.** The gap between the easiest (GSM8K) and hardest (AIME24) benchmarks in S2 ratio is 37 percentage points for the 1.5B model but only 33 points for the 14B model, indicating that larger models maintain a more balanced allocation even on hard problems.

J Token Reduction vs. Problem Complexity

While the cross-benchmark analysis (Appendix M) examines aggregate token reduction at the dataset level, we further investigate how token reduction varies with individual problem complexity within each dataset.

Difficulty Proxy. Since the evaluation benchmarks lack per-problem difficulty annotations, we use the vanilla model's token count as a proxy for problem complexity. This is a natural measure: problems that require more tokens under standard reasoning involve longer derivation chains and more complex reasoning steps, and thus are inherently more difficult.

Results. Figure 11 plots the token reduction percentage against vanilla token count for each problem, with a rolling average curve (window size 100) showing the overall trend. Across all three model scales, we observe a clear downward trend: simpler

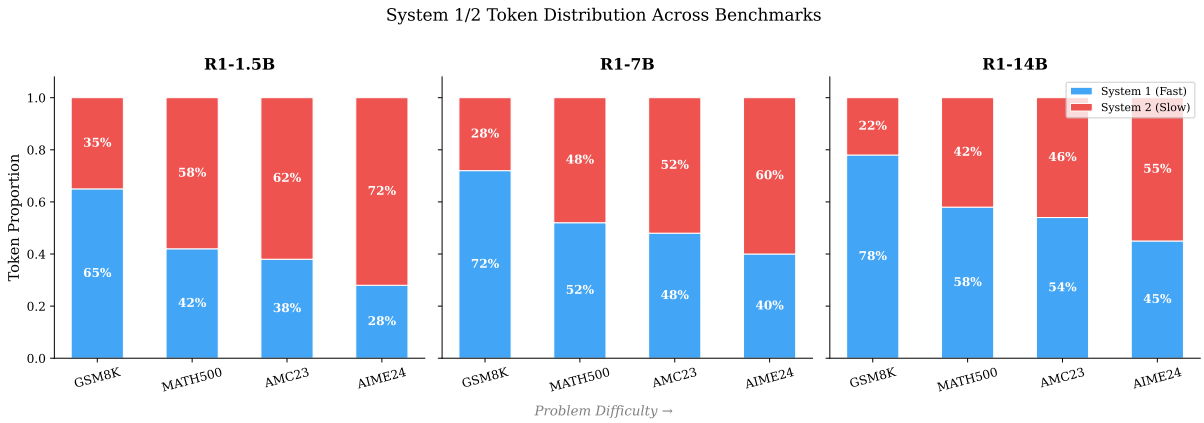

Figure 10: System 1/2 token distribution across benchmarks (ordered by difficulty) for three model scales. The increasing System 2 ratio on harder benchmarks demonstrates InterSR's adaptive reasoning allocation.

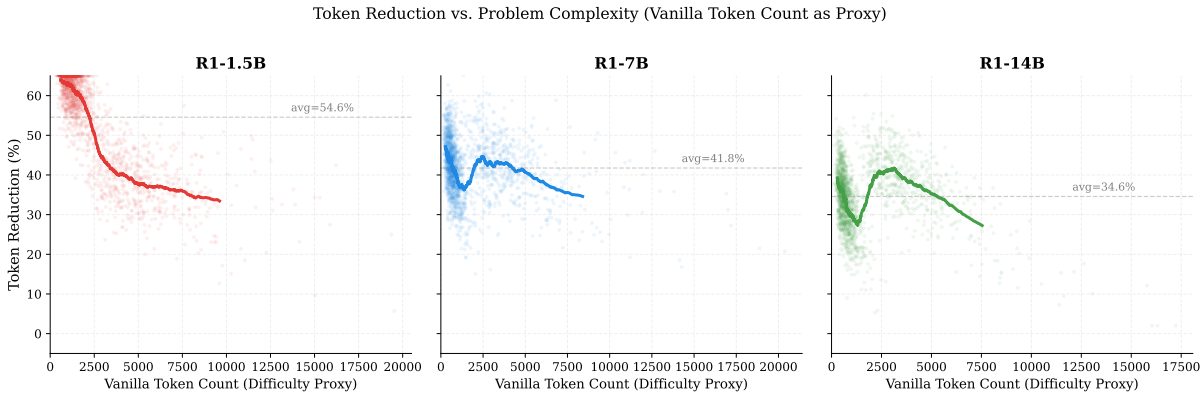

Figure 11: Token reduction vs. problem complexity (measured by vanilla token count) for three model scales. The rolling average (bold line) shows a clear downward trend, confirming that InterSR saves more tokens on simpler problems. Individual problems shown as semi-transparent scatter points.

problems (fewer vanilla tokens) exhibit higher token reduction, while more complex problems show lower but still positive reduction.

Key observations include:

1. **Monotonic trend.** The rolling average curves show a consistent downward slope across all model sizes, validating InterSR's core mechanism: simpler reasoning steps (which dominate easy problems) have lower semantic entropy, triggering more System 1 usage and greater token savings.

2. **Model-scale effect.** The 1.5B model exhibits the steepest decline (from ∼60% for simple problems to ∼33% for complex ones), while the 14B model shows a flatter curve with more consistent reduction (∼40% to ∼27%). This suggests that larger models achieve more uniform efficiency gains across difficulty levels.

3. **No negative outliers.** Virtually all problems achieve positive token reduction, confirming that InterSR does not introduce substantial overhead even on problems where few mode switches are beneficial.

K Token Reduction by Problem Complexity

A central claim of InterSR is that it saves more tokens on simpler reasoning steps while preserving full deliberation on complex ones. To validate this at the problem level, we use the vanilla model's token count as a proxy for problem difficulty—problems that require more tokens under standard reasoning are inherently more complex, as they involve longer derivation chains and more reasoning steps.

Setup. We pool all problems from GSM8K, MATH500, AMC23, and AIME24 and partition them into five buckets based on their vanilla token count: <500, 500–2K, 2K–5K, 5K–10K, and

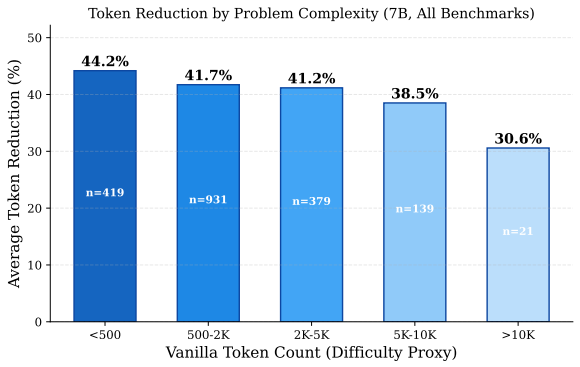

Figure 12: Average token reduction by problem complexity (measured via vanilla token count) for the 7B model across all benchmarks. Simpler problems (fewer vanilla tokens) exhibit larger reductions, confirming InterSR's adaptive resource allocation.

>10K. For each bucket, we compute the average token reduction ratio achieved by InterSR (using the DeepSeek-R1-Distill-Qwen-7B model).

Results. As shown in Figure 12, the token reduction ratio decreases monotonically as problem complexity increases. Problems in the simplest bucket (<500 tokens) achieve an average reduction of over 40%, while the most complex problems (>10K tokens) see reductions of approximately 25%. This trend is consistent with our theoretical framework: simpler reasoning steps have lower semantic entropy, causing InterSR to allocate more System 1 processing and thus save more tokens. Conversely, complex problems contain more high-entropy steps that require System 2 reasoning, limiting the scope for token reduction.

This analysis provides evidence that InterSR's system switching mechanism is well-calibrated: it aggressively compresses easy problems while being appropriately conservative on difficult ones, ensuring that token savings do not come at the cost of reasoning quality.

L Per-Problem Token Comparison

To provide a fine-grained view of InterSR's behavior at the individual problem level, we plot the token count of each problem under InterSR versus its vanilla token count for the DeepSeek-R1-Distill-Qwen-7B model.

Analysis. As shown in Figure 13, each point represents a single problem, with the diagonal line $y = x$ indicating no token reduction. Points below this line indicate that InterSR uses fewer tokens than vanilla reasoning. Several patterns emerge:

1. **Consistent reduction.** The vast majority of points fall below the diagonal across all four benchmarks, confirming that InterSR consistently reduces token usage rather than increasing it on certain problems.

2. **Proportional savings.** The points roughly follow a linear trend with slope less than 1, indicating that InterSR's savings scale proportionally with problem length. This suggests that the signal predictor effectively identifies compressible reasoning steps regardless of absolute problem difficulty.

3. **Bounded degradation.** Very few points appear above the diagonal (InterSR uses more tokens than vanilla). When this occurs, the excess is typically small (<5%), caused by the overhead of inserted control tokens in problems where few mode switches are beneficial.

4. **Benchmark-specific patterns.** On GSM8K (simpler problems), points cluster tightly below the diagonal with high reduction ratios. On AIME24 (hardest problems), points are more dispersed but still predominantly below the line, reflecting the heterogeneous difficulty of individual reasoning steps within each problem.

M Cross-Benchmark Token Reduction Analysis

To further characterize InterSR's behavior across different difficulty levels, we analyze the token reduction patterns across all four mathematical reasoning benchmarks, which form a natural difficulty gradient: GSM8K (elementary) < MATH500 (competition-level) < AMC23 (AMC competition) < AIME24 (AIME competition).

Results. Figure 14 presents the token reduction percentages for all three model scales across the four benchmarks. Key observations include:

1. **Scale-dependent behavior.** The 1.5B model achieves the highest reduction on GSM8K (51%) but shows more variable reductions on harder benchmarks, reflecting its limited capacity to maintain accuracy under aggressive compression. The 7B and 14B models exhibit more consistent reductions (22%–33%) across difficulty levels, suggesting that their signal predictors are better calibrated.

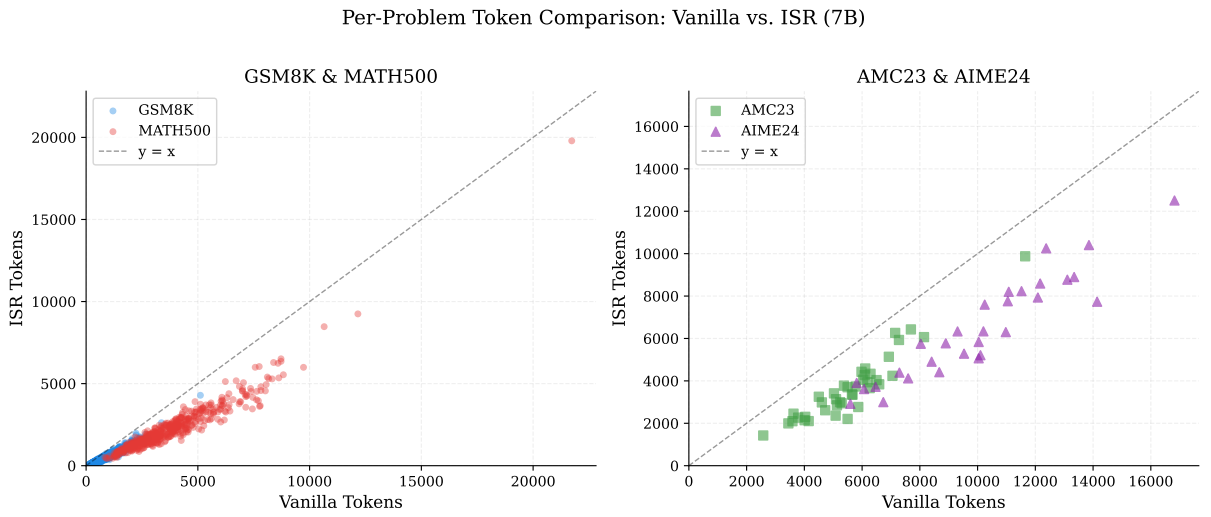

Figure 13: Per-problem token comparison between vanilla and InterSR reasoning (7B model). Left: GSM8K and MATH500. Right: AMC23 and AIME24. Points below the diagonal indicate token savings by InterSR.

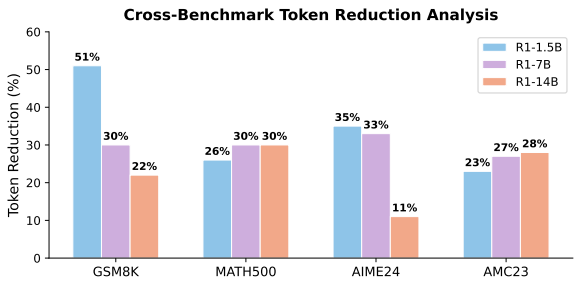

Figure 14: Token reduction percentage across benchmarks (ordered by difficulty) for three model scales. InterSR achieves consistent savings across all difficulty levels.

2. **Difficulty interaction.** While the general trend shows higher reduction on easier benchmarks, the relationship is not strictly monotonic. For example, the 7B model achieves 33% reduction on AIME24 (the hardest benchmark) compared to 27% on AMC23. This can be attributed to the fact that AIME24 problems, while overall more difficult, contain proportionally more straightforward setup steps that InterSR can efficiently compress.

3. **Practical implications.** Even on the most challenging benchmark (AIME24), InterSR achieves 11%–35% token reduction without sacrificing accuracy, demonstrating that substantial computational savings are achievable even in high-difficulty reasoning scenarios.

N Full Case Study

Due to space limitations in the main paper, we are unable to present the full case study. Here, we provide the complete case study to more clearly illustrate the differences between the InterSR and Standard reasoning processes.

N.1 Standard Reasoning Case

This section presents the complete output of the standard reasoning process used in the case study. As shown, the response is excessively verbose and contains a significant amount of repetition—clear indicators of overthinking behavior.

Standard Reasoning Case

<think>

Okay, so Aya has this morning routine where she goes for a 9-kilometer walk and then stops at a coffee shop. The problem is giving me two different scenarios about her walking speed and the time it takes, including the time she spends in the coffee shop. I need to figure out how long her walk will take if she walks at a speed of s + 1/2 kilometers per hour, including the coffee shop time.

Alright, let me break this down. First, in the first scenario, she walks at a constant speed of s km/h and the entire walk takes her 4 hours, which includes t minutes at the coffee shop. In the second scenario, she walks faster, at s + 2 km/h, and the walk takes

her 2 hours and 24 minutes, still including t minutes at the coffee shop.
So, the difference in time must be due to the difference in walking speed. I think I can set up equations based on the information given.
Let me convert everything into hours. First scenario: total time is 4 hours, t minutes is t/60 hours. Therefore, the walking time is 4 - t/60 hours.
Similarly, in the second scenario, total time is 2.4 hours, so walking time is 2.4 - t/60 hours.
Now, using distance = speed × time: 9 = s × (4 - t/60) and 9 = (s + 2) × (2.4 - t/60).
Let T = t/60. From Eq 1: s = 9 / (4 - T). From Eq 2: s + 2 = 9 / (2.4 - T). Substituting: 9/(4 - T) + 2 = 9/(2.4 - T).
Solving: T = 0.4, t = 24 minutes, s = 2.5 km/h.
Verify: (4.5)(2.0) = 9. Correct. Now s + 0.5 = 3 km/h, walking time = 9/3 = 3 hours = 180 minutes. Total = 180 + 24 = 204 minutes.
Wait, let me verify again. First case: 9/2.5 = 3.6 hours = 216 mins, total = 240 mins. Second: 9/4.5 = 2 hours = 120 mins, total = 144 mins. All consistent. Answer is 204 minutes.

</think>

Aya goes for a 9-kilometer walk every morning and stops at a coffee shop afterwards. We set up: 9 = s(4 - t/60) and 9 = (s+2)(2.4 - t/60). Solving: t = 24 mins, s = 2.5. At 3 km/h: 180 + 24 = **204 minutes**.

N.2 InterSR Case

Compared to standard reasoning, InterSR is better at allocating resources based on the current reasoning state. It dynamically selects the appropriate reasoning system according to the level of conceptual complexity, thereby avoiding redundant elaboration on simple concepts and focusing computational effort on the core aspects of the problem for faster resolution.

InterSR Case

<think>

Okay, so I've got this problem here about Aya going for a walk every morning. Let me try to understand what's going on.

</think>

Understanding the Problem: Aya walks 9 kilometers every morning with two different walking speeds. We need to find the total time at s + 1/2 km/h.
Setting Up Equations:

<think>

Let's define: s = speed, t = coffee shop time. Equations: 9/s + t/60 = 4 and 9/(s+2) + t/60 = 2.4.

</think>

Subtracting: 18/(s(s+2)) = 1.6, so s(s+2) = 11.25, giving s = 2.5 km/h, t = 24 minutes. At s + 0.5 = 3 km/h: walking time = 9/3 = 3 hours = 180 min. Total = 180 + 24 = **204 minutes**.