ADAPTIVE STREAMING REASONING WITH HIERARCHICAL RELATIVE POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models typically follow a *read-then-think* paradigm: they observe the complete input, reason over a static context, and then produce the answer. Yet many real-world scenarios involve continuously evolving inputs, such as audio and video streams, where latency-sensitive reasoning can benefit from processing information incrementally as it arrives rather than waiting for the entire input to complete. Recent streaming reasoning methods allow models to *think while reading*, but they largely rely on supervised imitation of pre-constructed trajectories, which limits their flexibility. In this paper, we propose **AdaSR**, an adaptive streaming reasoning framework that enables models to reason during input streaming and perform final deliberation once the stream is complete, learning when to think, and how much computation to allocate across different stages. To optimize this hierarchical reasoning process, we introduce **Hierarchical Relative Policy Optimization (HRPO)**, which decomposes policy optimization into streaming reasoning and deep reasoning phases, providing more *fine-grained advantage assignment* instead of uniformly distributing a single sequence-level advantage over all tokens. HRPO integrates format, accuracy, and adaptive thinking rewards to enforce valid reasoning protocols, preserve final task performance, and encourage latency-aware computation allocation. Experiments show that AdaSR achieves a better balance among reasoning accuracy, computational efficiency, and streaming latency compared with supervised fine-tuning baseline.

1 INTRODUCTION

Large reasoning models (OpenAI, 2024; DeepSeek-AI, 2025) have achieved remarkable performance on complex tasks such as mathematical reasoning, code generation, and multi-step decision making, often through chain-of-thought (CoT) reasoning (Wei et al., 2022): they receive an input, generate intermediate reasoning steps, and finally produce an answer. This paradigm follows a read then think pattern, where reasoning begins only after the model has observed a static and complete context. Real-world environments, however, are often dynamic. Inputs do not always appear as complete contexts; instead, they arrive as continuous streams: speech unfolds over time, videos reveal information frame by frame, interactive agents receive observations sequentially, and sensor-driven systems must react before an event is fully observed (Gu et al., 2017; Ma et al., 2018; Arivazhagan et al., 2019; Chen et al., 2024; Lin et al., 2024; Défossez et al., 2024). In such settings, waiting for the full input introduces unnecessary latency, whereas reasoning too early may lead to premature or misleading conclusions. This raises a fundamental question: how can reasoning models think, update, and respond under continuously evolving observations?

Recent efforts have begun to explore streaming reasoning, enabling models to reason in real time as inputs dynamically unfold (Tong et al., 2025a; Zhang et al., 2026). However, existing streaming CoT methods typically rely on supervised fine-tuning over carefully constructed streaming trajectories. Such fine-grained supervision is costly, since each partial observation may require a corresponding local reasoning annotation. More importantly, imitation learning tends to encourage models to reproduce the surface form of streaming traces, rather than acquire the underlying ability to decide whether a partial input requires shallow understanding, deeper reasoning, or no reasoning at all. As a result, models trained on fixed supervised trajectories may exhibit the appearance of streaming reasoning, yet still lack genuine adaptivity.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

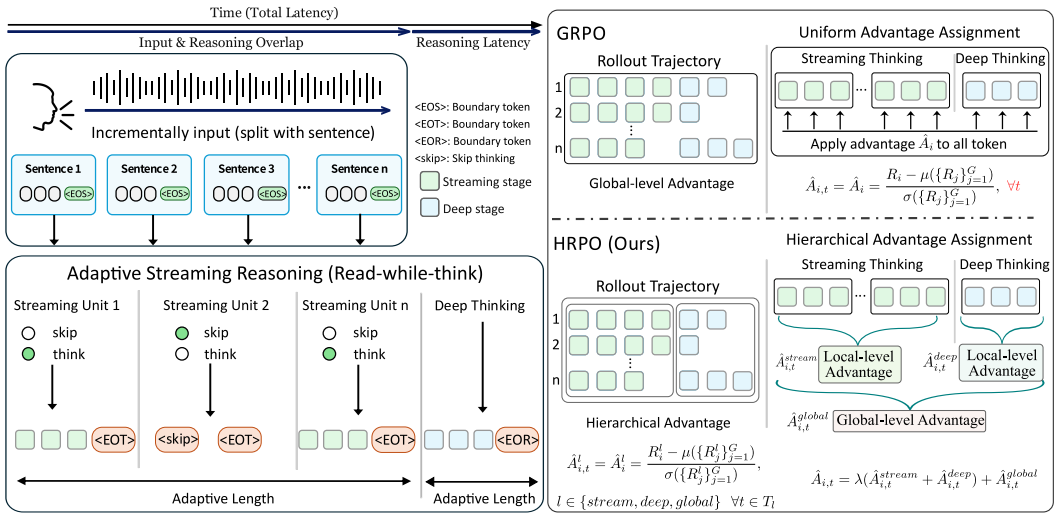

Figure 1: Overview of the AdaSR framework. *(left)* AdaSR thinks while reading streaming inputs, decide whether to reason over each segment, and allocate computation between streaming thinking and final deep thinking. *(right)* AdaSR introduces HRPO to align optimization with the temporal structure of streaming reasoning, assigning fine-grained advantages across local and global scopes.

To move beyond imitation-based streaming reasoning, we propose AdaSR, an adaptive streaming reasoning framework that leverages reinforcement learning to optimize how models reason over evolving inputs, as shown in Figure 1. AdaSR learns a computation policy during input streaming: when to reason, when to skip, and how to balance streaming reasoning with final deliberation, echoing broader efforts on adaptive computation and test-time compute allocation (Graves, 2016; Snell et al., 2024; Muennighoff et al., 2025). A natural starting point is Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which has shown strong effectiveness in improving reasoning models with outcome-based rewards (Ziegler et al., 2019; Ouyang et al., 2022; Cobbe et al., 2021; Shao et al., 2024; DeepSeek-AI, 2025). However, streaming reasoning presents a structured optimization problem that standard GRPO does not explicitly capture. A streaming trajectory consists of multiple reasoning units generated under partial observations, followed by a deep reasoning phase that integrates the complete context. Assigning a single sequence-level advantage uniformly to all generated tokens blurs this temporal structure, making it difficult to distinguish the contributions of local understanding, global integration, and final correctness.[1]

To address this issue, AdaSR introduces Hierarchical Relative Policy Optimization (HRPO), which refines GRPO with stage-aware advantage assignment. Instead of treating the entire trajectory as a flat sequence, HRPO decomposes the learning signal according to the temporal structure of streaming reasoning, assigning distinct advantages to streaming tokens, deep-reasoning tokens, and the full rollout. This preserves GRPO's group-relative optimization while turning the coarse sequence-level advantage into finer-grained, hierarchy-aware credit signals. Combined with format, accuracy, and adaptive thinking rewards, AdaSR optimizes both final task performance and the computation path that leads to it. Experiments show that AdaSR achieves a better balance among reasoning accuracy, computational efficiency, and latency compared with supervised fine-tuning baselines.

Our contributions are fourfold: **(1)** We propose AdaSR, to the best of our knowledge the first adaptive streaming reasoning framework that learns when to reason, when to skip, and how to allocate computation over evolving inputs, guided by adaptive thinking rewards. **(2)** We introduce HRPO, a hierarchy-aware advantage assignment method that replaces coarse sequence-level credit with local-to-global advantages, enabling fine-grained optimization for structured streaming reasoning trajectories. **(3)** We adapt streaming reasoning to a vLLM-compatible rollout and inference pipeline (Kwon et al., 2023), improving its practicality for efficient training and deployment in realistic streaming scenarios. **(4)** Experiments show that AdaSR achieves a better trade-off among reasoning accuracy and latency than supervised fine-tuning baselines.

[1]GRPO's uniform advantage assignment leads to a cross-phase credit paradox: deep-phase success can falsely amplify redundant streaming thoughts, while deep-phase inefficiency can unfairly suppress useful ones.

2 PRELIMINARY

Streaming Thinking The standard reasoning paradigm follows a *read-then-think* pattern: the model receives the full context before reasoning. Given a question Q and a context $C = \{C_1, \ldots, C_T\}$ decomposed into sequential sentences, with R_t denoting the reasoning segment associated with the prefix ending at C_t, this component can be summarized as $\mathcal{P}_{\text{standard}} = \prod_{t=1}^{T} P(R_t|Q, C_{\leq T}, R_{\leq t-1})$. In contrast, *streaming thinking* (Tong et al., 2025a; Zhang et al., 2026; Chiang et al., 2025a) lets reasoning unfold with the input stream: as each sentence arrives, the model forms a local thought under partial context, then performs a final deep deliberation once the complete context and task instruction I are available:

$$\mathcal{P}_{\text{streaming}} = \underbrace{P(R_q|Q) \cdot \prod_{t=1}^{T} P(R_t|Q, C_{\leq t}, R_{\leq t-1})}_{\text{streaming thinking}} \cdot \underbrace{P(R|Q, C_{\leq T}, R_{\leq T})}_{\text{deep thinking}}. \tag{1}$$

The first factor is the streaming phase: after each sentence C_t, the model emits R_t before seeing future sentences, or skips irrelevant input. Boundary and end-of-thought tokens separate local segments under streaming masks. The second factor is the deep phase, integrating $C_{\leq T}$ and $R_{\leq T}$ into the final response R. The phase boundary is determined separately for each trajectory by its number of generated streaming tokens.

GRPO for Reasoning Reinforcement learning improves LLM reasoning by optimizing outcome rewards without explicit chain-of-thought annotations (DeepSeek-AI, 2025). In PPO (Schulman et al., 2017), old-policy rollouts are reweighted token by token and clipped to stabilize updates, while advantages are typically estimated by a learned value function. GRPO (Shao et al., 2024) keeps this clipped update but removes the value model: for a question-answer pair $(q, a) \sim \mathcal{D}$, the old policy $\pi_{\theta_{\text{old}}}$ samples G candidate outputs $\{o_i\}_{i=1}^{G}$, whose rewards are compared within the same group to form relative advantages. The objective averages the clipped token-level surrogate over each candidate sequence while keeping π_θ close to π_{ref}:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{(q,a)\sim\mathcal{D} \\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}} \left\{ \frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left[\mathcal{C}\big(r_{i,t}(\theta), \hat{A}_i\big) - \beta\, \mathrm{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big) \right] \right\}. \tag{2}$$

where the clipped term $\mathcal{C}\big(r_{i,t}(\theta), \hat{A}_i\big) = \min\big(r_{i,t}(\theta)\hat{A}_i, clip(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_i\big)$ inherits PPO's trust-region behavior through the clipping threshold ε, with β controlling the KL regularization strength. In this term, the token-wise importance ratio is paired with a sequence-level advantage obtained by normalizing the reward $R_i = R(q, a, o_i)$ against the G rewards sampled for the same question:

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}, \qquad \hat{A}_i = \frac{R_i - \mu(\{R_j\}_{j=1}^{G})}{\sigma(\{R_j\}_{j=1}^{G})}. \tag{3}$$

Since the same $\hat{A}_i$ is attached to every token position t in o_i, GRPO treats a candidate output as one flat trajectory. This is suitable for batch reasoning, where the model reasons after observing the full input, but it is too coarse for streaming reasoning: local thoughts produced under partial observations and final deliberation over the complete context receive indistinguishable credit.

3 METHODOLOGY

We present Adaptive Streaming Reasoning(**AdaSR**), an RL framework for adaptive computation allocation between streaming and deep reasoning. As shown in Figure 1, AdaSR combines hierarchical relative policy optimization (HRPO §3.1), adaptive rewards (§3.2), and streaming rollout (§3.3).

3.1 HIERARCHICAL RELATIVE POLICY OPTIMIZATION

The GRPO assigns a uniform advantage to all tokens in a trajectory, treating it as a flat sequence. For trajectory o_i, let L_s^i and L_d^i denote the numbers of generated streaming and deep tokens, respectively, with $|o_i| = L_s^i + L_d^i$. Streaming reasoning trajectories have an inherently *hierarchical* structure: the streaming phase ($1 \leq t \leq L_s^i$) produces local and partial observations, while the deep phase ($L_s^i < t \leq |o_i|$) integrates the full context. The **HRPO** decomposes advantage estimation into fine-grained levels.

Hierarchical Advantage Assignment The central difficulty in streaming reasoning is temporal credit assignment: streaming tokens should be credited for local decisions under partial observations, deep tokens for final integration, and the whole trajectory for answer correctness. HRPO keeps the group-relative normalization of GRPO, but separates these signals into streaming-local (s), deep-local (d), and trajectory-global (g) advantages.

For each question q, HRPO samples G trajectories $\{o_i\}_{i=1}^{G}$ from $\pi_{\theta_{\text{old}}}$. Let A_i^s, A_i^d, and A_i^g denote the group-relative advantages associated with the streaming-local, deep-local, and trajectory-global levels, respectively. HRPO defines hierarchical advantages by attaching the appropriate level-wise advantage to the corresponding token range:

$$\begin{aligned} &\hat{A}_{i,t}^{\ell} = A_i^{\ell}, \quad t \in \mathcal{T}_i^{\ell}, \quad \ell \in \{s, d, g\}, \\ &\mathcal{T}_i^{s} = [1, L_s^i], \quad \mathcal{T}_i^{d} = (L_s^i, |o_i|], \quad \mathcal{T}_i^{g} = [1, |o_i|]. \end{aligned} \tag{4}$$

where the intervals in $\mathcal{T}_i^{\ell}$ denote integer token positions, so $L_s^i = |\mathcal{T}_i^s|$ and $L_d^i = |\mathcal{T}_i^d| = |o_i| - L_s^i$. For each level $\ell \in \{s, d, g\}$, HRPO uses the shared per-token importance ratio $r_{i,t}^{\ell}(\theta) = \pi_\theta(o_{i,t}|q, o_{i,<t}) / \pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})$. The superscript records which hierarchical advantage and surrogate term the ratio is paired with. Thus, local advantages shape phase-specific behavior, while the global advantage preserves answer-level correctness across the full rollout.

Policy Optimization Objective Based on the fine-grained advantages and ratios above, HRPO defines three clipped surrogate losses over their corresponding token ranges. The objective is:

$$\begin{aligned} J_{\text{HRPO}}(\theta) = \mathbb{E}_{\substack{(q,a)\sim\mathcal{D} \\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}}} \Bigg\{ \frac{1}{G}\sum_{i=1}^{G} \Bigg[\lambda \underbrace{\Bigg(\frac{1}{L_s^i}\sum_{t=1}^{L_s^i} \mathcal{C}\big(r_{i,t}^{s}, \hat{A}_{i,t}^{s}\big) + \frac{1}{L_d^i}\sum_{t=L_s^i+1}^{|o_i|} \mathcal{C}\big(r_{i,t}^{d}, \hat{A}_{i,t}^{d}\big) \Bigg)}_{\text{local clipped surrogate objective}} \\ + \underbrace{\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \mathcal{C}\big(r_{i,t}^{g},\, \hat{A}_{i,t}^{g}\big)}_{\text{global clipped surrogate objective}} - \frac{\beta}{|o_i|}\sum_{t=1}^{|o_i|} \mathrm{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big) \Bigg] \Bigg\}, \end{aligned} \tag{5}$$

where $\mathcal{C}(r_{i,t}^{\ell}, \hat{A}_{i,t}^{\ell}) = \min\big(r_{i,t}^{\ell}\,\hat{A}_{i,t}^{\ell},\ \text{clip}(r_{i,t}^{\ell},\, 1{-}\varepsilon, 1{+}\varepsilon)\,\hat{A}_{i,t}^{\ell}\big)$ for $\ell \in \{s, d, g\}$ is the clipped surrogate objective (Schulman et al., 2017).[2]

The coefficient $\lambda \in [0, 1]$ balances local and global objectives. When $\lambda = 0$, HRPO falls back to the global trajectory-level signal; increasing λ strengthens phase-specific optimization. Unlike the GRPO which applies one advantage uniformly across the entire trajectory, HRPO applies differentiated local advantages to each phase while simultaneously maintaining a global correctness signal across all tokens, resolving the temporal credit assignment problem inherent in streaming reasoning.

3.2 Adaptive Streaming Reasoning Reward

In streaming reasoning, the model must reason incrementally as the input unfolds, deciding not only what intermediate thoughts to produce but also how much computation to allocate at each stage. Optimizing only for terminal accuracy provides no direct supervision over this computation path, and may therefore encourage inefficient trajectories, such as overly verbose streaming thoughts or delayed reasoning concentrated in the deep phase. To guide both task performance and computation allocation, we decompose the reward into format, accuracy, and adaptive thinking components, corresponding to structural validity, final correctness, and latency-aware reasoning efficiency.

Format Reward The format reward makes rollouts parseable for HRPO. During streaming, each segment must terminate with `<EOT>`, and the content before `<EOT>` must be either a reasoning thought or `<skip>`. During deep reasoning, the output must contain a non-empty deep reasoning field ending with `<EOR>`. For rollout i, we denote the corresponding binary rewards by $R_{i,\text{fmt}}^{s}$ and $R_{i,\text{fmt}}^{d}$. This component constrains structural validity rather than reasoning quality.

[2] We provide the policy gradient analysis in Appendix B.

Accuracy Reward The accuracy reward provides the terminal task signal: $R_i^{\text{acc}} = \mathbb{1}[\hat{a}_i = a_i]$, where $\hat{a}_i$ is the model prediction and a_i is the ground-truth answer. It evaluates the complete trajectory after both streaming and deep reasoning, anchoring learning to final correctness.

Adaptive Thinking Reward The adaptive thinking reward uses token length as a proxy for computation allocation. It discourages excessive computation while allowing the policy to allocate more tokens when additional reasoning is useful. We first define phase-local length-shaping rewards:

$$R_i^{L_s} = -\log\left(1 + L_s^i\right), \quad R_i^{L_d} = -\log\left(1 + L_d^i\right), \tag{6}$$

where $R_i^{L_s}$ and $R_i^{L_d}$ are local length rewards for streaming and deep reasoning, respectively. The logarithmic form discourages verbosity with diminishing marginal strength.

Local rewards alone do not capture the latency asymmetry between phases: streaming reasoning can overlap with input arrival, whereas deep reasoning begins only after the stream ends. We therefore introduce a success-conditioned trajectory-level efficiency reward:

$$R_i^{\text{eff}} = \underbrace{R_i^{\text{acc}} R_{i,\text{fmt}}^s R_{i,\text{fmt}}^d}_{\text{acc. \& format gate}} \underbrace{\exp\left[-\frac{\alpha L_s^i + L_d^i}{\tau}\right]}_{\text{latency-aware allocation}}. \tag{7}$$

where $0 < \alpha < 1$ and $\tau > 0$ control the latency discount and shared reward scale, respectively.[3] The exponential term then rewards trajectories with lower effective latency cost $\alpha L_s^i + L_d^i$. Thus, the global reward favors trajectories that are correct, parseable, and computationally sufficient.

Hierarchical Advantage Composition Eq. 4 specifies where each level-wise advantage is applied: A_i^s is assigned to streaming tokens, A_i^d to deep-reasoning tokens, and A_i^g to the full trajectory. Following the multi-reward normalization observation in GDPO (Liu et al., 2026a), we avoid first summing weighted rewards and then normalizing the result, which can blur distinct reward components into the same advantage signal. Instead, for any component reward X_i within the G rollouts of the same question, we first compute the component advantage $\mathcal{N}_{\mathcal{G}}(X_i) = \big(X_i - \mu(\{X_j\}_{j=1}^G)\big)/\sigma(\{X_j\}_{j=1}^G)$, and then compose advantages at their corresponding temporal levels:

$$A_i^s = \mathcal{N}_{\mathcal{G}}(R_{i,\text{fmt}}^s) + \eta\mathcal{N}_{\mathcal{G}}(R_i^{L_s}), \; A_i^d = \mathcal{N}_{\mathcal{G}}(R_{i,\text{fmt}}^d) + \eta\mathcal{N}_{\mathcal{G}}(R_i^{L_d}), \; A_i^g = \mathcal{N}_{\mathcal{G}}(R_i^{\text{acc}}) + \eta\mathcal{N}_{\mathcal{G}}(R_i^{\text{eff}}). \tag{8}$$

This advantage-level composition matches Eq. 4: format and length signals supervise the corresponding streaming or deep time span, while accuracy and efficiency provide a trajectory-level signal. The shared η controls the strength of the adaptive thinking signal after normalization. [4]

3.3 Training Algorithm

Streaming Rollout Following StreamingThinker (Tong et al., 2025a), we implement streaming rollout by extending the vLLM inference engine (Kwon et al., 2023). At the start of each training iteration, we set $\theta_{\text{old}} \leftarrow \theta$ and hold this behavior policy fixed during rollout and the corresponding actor updates. Input sentences are fed sequentially: after prefilling each sentence C_t, the model decodes streaming reasoning tokens R_t until `<EOT>`. Position IDs implement streaming position encoding—input tokens and reasoning tokens maintain independent positional indices (Su et al., 2024; Tong et al., 2025a). Streaming attention masks prevent attending to future input during the streaming phase. After all T sentences, the instruction I is appended and the model generates deep reasoning R. The trajectory $o_i = [R_1^{(i)}, \ldots, R_T^{(i)}, R^{(i)}]$ is collected with per-token log-probabilities.

Policy Gradient Computation Given the collected trajectories, we compute log-probabilities under π_θ using streaming attention masks. The reward components are computed, converted into group-relative component advantages, composed into level-wise advantages, and assigned to token ranges. We average the sampled HRPO objectives (Eq. 5) across questions in the minibatch and

[3] α discounts streaming tokens because they can overlap with input arrival, while deep tokens incur full post-stream latency. We provide the details in Appendix C.

[4] If η were instead applied at the reward level before group normalization, then in the active gated regime the positive scale would be canceled and η would lose its effect; we provide the proof in Appendix C.

Table 1: Main results on streaming reasoning benchmarks with Qwen3 models. The offline baseline uses read-then-think; StreamingThinker denote the SFT baseline; GRPO and HRPO denote AdaSR-GRPO and AdaSR-HRPO (ours). Stream, Deep, and Total denote streaming-stage, deep-stage, and total token lengths. Percentages denote relative changes over StreamingThinker; red indicates improvements and green degradations.

	GSM-symbolic P1				GSM-symbolic P2			
Metric	offline	StreamingThinker	GRPO	HRPO	offline	StreamingThinker	GRPO	HRPO
Qwen3-1.7B								
Acc.↑	0.634	0.834	0.850 (+1.9%)	**0.871 (+4.4%)**	0.424	0.642	0.758 (+18.1%)	**0.788 (+22.7%)**
Stream↓	–	165.71	172.99 (+4.4%)	**160.03 (-3.4%)**	–	229.29	235.78 (+2.8%)	**210.05 (-8.4%)**
Deep↓	–	**118.04**	133.80 (+13.3%)	131.97 (+11.8%)	–	**132.62**	148.70 (+12.1%)	160.21 (+20.8%)
Total↓	1598.70	**283.75**	306.79 (+8.1%)	292.00 (+2.9%)	1866.47	**361.91**	384.49 (+6.2%)	370.26 (+2.3%)
Qwen3-4B								
Acc.↑	0.636	0.934	0.946 (+1.3%)	**0.950 (+1.7%)**	0.462	0.816	0.874 (+7.1%)	**0.884 (+8.3%)**
Stream↓	–	167.79	157.31 (-6.2%)	**150.69 (-10.2%)**	–	227.21	209.87 (-7.6%)	**196.39 (-13.6%)**
Deep↓	–	119.21	**111.15 (-6.8%)**	116.56 (-2.2%)	–	130.37	**126.27 (-3.1%)**	131.16 (+0.6%)
Total↓	1717.16	286.99	268.45 (-6.5%)	**267.25 (-6.9%)**	1937.40	357.58	336.13 (-6.0%)	**327.55 (-8.4%)**
Qwen3-8B								
Acc.↑	0.691	0.946	0.958 (+1.3%)	**0.970 (+2.5%)**	0.559	0.856	0.899 (+5.0%)	**0.923 (+7.8%)**
Stream↓	–	162.75	149.44 (-8.2%)	**135.62 (-16.7%)**	–	218.12	197.27 (-9.6%)	**170.86 (-21.7%)**
Deep↓	–	114.44	**107.81 (-5.8%)**	109.57 (-4.3%)	–	126.46	123.74 (-2.2%)	**123.29 (-2.5%)**
Total↓	1631.30	277.19	257.25 (-7.2%)	**245.18 (-11.5%)**	1859.90	344.59	321.02 (-6.8%)	**294.15 (-14.6%)**
	MetaMathQA				PubMedQA			
Metric	offline	StreamingThinker	GRPO	HRPO	offline	StreamingThinker	GRPO	HRPO
Qwen3-1.7B								
Acc.↑	0.848	0.688	0.823 (+19.6%)	**0.826 (+20.1%)**	0.588	0.749	0.776 (+3.6%)	**0.793 (+5.9%)**
Stream↓	–	149.20	158.71 (+6.4%)	173.25 (+16.1%)	–	414.74	433.57 (+4.5%)	**414.21 (-0.1%)**
Deep↓	–	121.47	159.29 (+31.1%)	**129.36 (+6.5%)**	–	**120.53**	143.31 (+18.9%)	136.64 (+13.4%)
Total↓	1281.77	270.66	318.00 (+17.5%)	**302.60 (+11.8%)**	434.63	535.27	576.89 (+7.8%)	**550.85 (+2.9%)**
Qwen3-4B								
Acc.↑	0.828	0.860	0.909 (+5.7%)	**0.924 (+7.4%)**	0.615	0.756	0.792 (+4.8%)	**0.807 (+6.7%)**
Stream↓	–	152.20	140.01 (-8.0%)	**121.71 (-20.0%)**	–	402.88	378.21 (-6.1%)	**356.73 (-11.5%)**
Deep↓	–	122.00	**113.55 (-6.9%)**	115.65 (-5.2%)	–	116.11	**111.84 (-3.7%)**	114.51 (-1.4%)
Total↓	1379.09	274.20	253.56 (-7.5%)	**237.36 (-13.4%)**	331.84	518.98	490.06 (-5.6%)	**471.24 (-9.2%)**
Qwen3-8B								
Acc.↑	0.849	0.885	0.925 (+4.5%)	**0.951 (+7.5%)**	0.661	0.785	0.817 (+4.1%)	**0.846 (+7.8%)**
Stream↓	–	149.16	131.61 (-11.8%)	**108.32 (-27.4%)**	–	390.79	359.30 (-8.1%)	**313.92 (-19.7%)**
Deep↓	–	115.90	109.00 (-5.9%)	**107.56 (-7.2%)**	–	111.46	**108.49 (-2.7%)**	108.79 (-2.4%)
Total↓	1351.51	265.05	240.61 (-9.2%)	**215.88 (-18.6%)**	**311.93**	502.25	467.79 (-6.9%)	**422.71 (-15.8%)**

update θ by gradient ascent. The reference policy π_{ref} is initialized from the starting SFT policy and remains frozen throughout training. [5]

4 Experiments

Experimental Setup We evaluate AdaSR with Qwen3 series models (Yang et al., 2025a) on reasoning tasks covering mathematical reasoning, context-based question answering, and logical reasoning. For in-domain evaluation, we use GSM-Symbolic (Mirzadeh et al., 2024), MetaMathQA (Yu et al., 2023), and PubMedQA (Jin et al., 2019); for out-of-domain evaluation, we further use GSM-Infinite (Zhou et al., 2025) and LogicNLI dataset (Tian et al., 2021). These datasets contain sufficiently long problem statements, contextual passages, or multi-step constraints that can be naturally segmented into sentence-level streams, making them suitable for evaluating reasoning under partial and progressively arriving inputs.

We evaluate AdaSR in terms of reasoning performance and streaming efficiency. Accuracy is the primary metric, reflecting the goal of improving reasoning under streaming inputs. We further

[5] Appendix E provides implementation details on the streaming vLLM rollout, attention masks, stream-aware logits computation, and veRL adaptation. The training process is shown in Algorithm 1.

Table 2: Effect of hierarchical advantage assignment on GSM-symbolic P2/P1. We report accuracy and phase-specific lengths for GRPO, HRPO, sentence-level, and token-level variants. Percentages are relative to GRPO; red/green denote improvements/degradations. Token-level assignment only affects the format boundary token.

Qwen3-1.7B	GSM-symbolic P2				GSM-symbolic P1			
Method	Acc↑	Streaming Len.↓	Deep Len.↓	Total Len.↓	Acc↑	Streaming Len.↓	Deep Len.↓	Total Len.↓
GRPO	0.758	235.784	148.704	384.488	0.850	172.994	133.798	306.792
HRPO	**0.788** (+4.0%)	**210.046** (-10.9%)	160.210 (+7.7%)	370.256 (-3.7%)	**0.871** (+2.5%)	160.032 (-7.5%)	131.972 (-1.4%)	292.004 (-4.8%)
HRPO-sentence	0.754 (-0.5%)	250.946 (+6.4%)	**131.024** (-11.9%)	381.970 (-0.7%)	0.868 (+2.1%)	180.677 (+4.4%)	**115.770** (-13.5%)	296.447 (-3.4%)
HRPO-token	0.770 (+1.6%)	213.528 (-9.4%)	145.432 (-2.2%)	**358.960** (-6.6%)	0.862 (+1.4%)	**158.733** (-8.2%)	120.714 (-9.8%)	**279.447** (-8.9%)

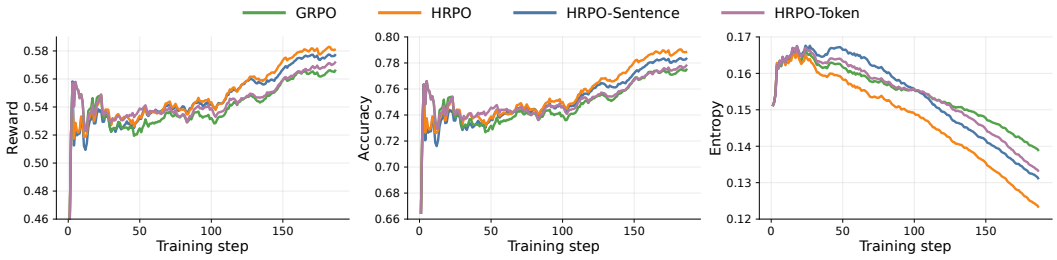

Figure 2: Training dynamics of GRPO and HRPO variants. From left to right, we show reward, answer accuracy, and policy entropy across training steps.

measure efficiency using token length and real latency.[6] Token length includes total generation length as well as streaming- and deep-reasoning lengths, while latency captures response delay under streaming inference, where streaming reasoning can overlap with input reception but deep reasoning directly delays the final answer.

Main Experiment Table 1 shows that AdaSR-HRPO improves streaming reasoning by better allocating computation rather than simply generating longer reasoning traces. Compared with the SFT-based StreamingThinker baseline, HRPO consistently improves accuracy across Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, with particularly large gains on harder benchmarks such as GSM-symbolic P2 and MetaMathQA. On Qwen3-1.7B, HRPO improves accuracy by $+22.7\%$ on GSM-symbolic P2 and $+20.1\%$ on MetaMathQA, suggesting that supervised streaming trajectories alone are insufficient for robust reasoning under partial observations.

Compared with GRPO, HRPO achieves a better accuracy-efficiency frontier: across all twelve settings, it improves accuracy while reducing total generation length. This indicates that hierarchical advantage assignment alleviates the tendency of sequence-level optimization to reward globally successful but locally inefficient traces. By separating credit across streaming and deep stages, HRPO encourages useful intermediate reasoning while suppressing redundant computation.

HRPO's joint accuracy-efficiency gains strengthen with model scale. Relative to StreamingThinker, the 1.7B model improves accuracy with increased total length, whereas the 4B and 8B models achieve higher accuracy with 6.9%–13.4% and 11.5%–18.6% fewer tokens, respectively. From 4B to 8B, HRPO's accuracy and token-efficiency advantages over GRPO widen on every benchmark, suggesting that greater model capacity helps exploit hierarchical credit assignment.

AdaSR's advantage is also better understood from a latency-aware perspective rather than raw total length alone. Unlike read-then-think methods, which defer all reasoning until the full input is observed, AdaSR shifts part of the computation into the streaming stage and keeps final deliberation compact. For example, on GSM-symbolic P2 with Qwen3-1.7B, HRPO reduces deep-stage reasoning from 1866.474 to 160.210 tokens while improving accuracy from 0.424 to 0.788, showing that AdaSR learns to reason while reading instead of postponing computation to the final response.

Influence of Hierarchical Advantage Assignment Table 2 examines how the granularity of advantage assignment affects streaming reasoning. GRPO applies a single sequence-level advantage to all generated tokens, while HRPO separates credit assignment into streaming and deep reasoning

[6] See Appendix F for the definition of real latency.

Table 3: Analysis of adaptive streaming rewards. We compare GRPO and HRPO under progressively structured rewards, from accuracy-only to format- and length-aware designs. Percentages in *Acc + format + length* are relative to *Acc + format*, with higher accuracy and lower lengths treated as improvements.

Qwen3-1.7B		GRPO			HRPO		
Benchmark	Metric	Acc reward	Acc + format	Acc + format + length	Acc reward	Acc + format	Acc + format + length
GSM-symbolic P2	Acc↑	0.726	**0.762**	0.756 (-0.8%)	0.726	0.778	**0.788** (+1.3%)
	Streaming Len.↓	**231.966**	287.526	235.784 (-18.0%)	231.966	275.562	**210.046** (-23.8%)
	Deep Len.↓	165.942	191.756	**148.704** (-22.5%)	165.942	160.990	**160.210** (-0.5%)
	Total Len.↓	397.908	479.282	**384.488** (-19.8%)	397.908	436.552	**370.256** (-15.2%)
GSM-symbolic P1	Acc↑	0.850	**0.875**	0.867 (-0.9%)	0.867	0.869	**0.871** (+0.2%)
	Streaming Len.↓	**168.723**	206.503	172.994 (-16.2%)	168.723	197.902	**160.032** (-19.1%)
	Deep Len.↓	137.984	161.667	**133.798** (-17.2%)	137.984	137.218	**131.972** (-3.8%)
	Total Len.↓	**306.707**	368.170	306.792 (-16.7%)	306.707	335.120	**292.004** (-12.9%)

Table 4: OOD performance comparison on GSM-Infinite (within-task distribution shift) and LogicNLI (cross-task generalization). We report accuracy and total generated tokens.

	Qwen3-4B				*Qwen3-8B*			
	GSM-Infinite		LogicNLI		GSM-Infinite		LogicNLI	
Method	Acc↑	Total Len.↓	Acc↑	Total Len.↓	Acc↑	Total Len.↓	Acc↑	Total Len.↓
SFT-based	0.479	273.408	0.474	**564.248**	0.519	259.738	0.514	536.036
GRPO	0.509	285.38	0.445	595.263	0.554	262.550	0.490	547.642
HRPO	**0.546**	**265.65**	**0.489**	591.985	**0.606**	**233.772**	**0.549**	**520.947**

stages. We further compare two finer-grained variants: HRPO-sentence, which assigns streaming advantages at the sentence level, and HRPO-token, which applies token-level assignment only to the boundary token associated with the format reward. Other reward components are not token-local by nature, and therefore follow the same assignment scheme as HRPO-sentence.

The results show that stage-level HRPO achieves the best overall accuracy-efficiency trade-off. Compared with GRPO, HRPO improves accuracy on both GSM-symbolic P2 and P1 while reducing total generation length, indicating that separating streaming and deep-stage credit can effectively mitigate the coarse credit assignment of standard GRPO. In contrast, finer-grained assignment does not consistently improve performance. HRPO-sentence reduces deep reasoning length but lowers accuracy, suggesting that sentence-level credit may over-localize reasoning signals and weaken cross-sentence reasoning coherence.

HRPO-token achieves the shortest total length, but its accuracy is lower than HRPO. This indicates that token-level assignment is useful for precise format control, such as boundary-token supervision, but is insufficient for assigning semantic reasoning credit. Overall, the results suggest that the most effective granularity is not the finest one, but the one that matches the natural structure of streaming reasoning: stage-level assignment between streaming and deep deliberation.

Figure 2 shows training dynamics that are consistent with the final evaluation. The left panel reports the total training reward, which combines accuracy, format, and length rewards, while the middle panel tracks answer accuracy. Stage-level HRPO achieves the strongest reward curve and the best final accuracy among the compared variants, indicating that separating credit between streaming and deep reasoning provides a more effective optimization signal. In contrast, the finer-grained variants do not consistently improve the learning curves, suggesting that overly localized assignment may fragment the semantic credit needed for reasoning. The entropy curves show an initial exploration phase followed by a steady decrease, indicating that the policies gradually become more confident as training progresses.

Analysis of Adaptive Streaming Rewards Table 3 examines how reward components affect adaptive streaming reasoning, where the model must decide when to reason or skip during the input stream. With accuracy reward alone, the model receives only a final outcome signal and lacks guidance on streaming-format compliance. Adding the format reward improves accuracy under both GRPO and HRPO, suggesting that parseable read-think trajectories stabilize training, though with longer generations.

Table 5: The throughput (token/s) and reasoning latency (s) under different inference backends. We compare read-then-think, StreamingThinker, AdaSR-GRPO, and AdaSR-HRPO using Transformers and vLLM backends. Throughput is higher-is-better, while latency is lower-is-better. Speedups in parentheses for latency denote reductions relative to read-then-think under the same backend.

Backend	Read-then-think		StreamingThinker		AdaSR-GRPO		AdaSR-HRPO	
	Throughput↑	Latency↓	Throughput↑	Latency↓	Throughput↑	Latency↓	Throughput↑	Latency↓
Transformers	33.170	56.250	31.810	**6.080** (9.25×)	31.450	6.710 (8.38×)	32.570	6.520 (8.63×)
vLLM	140.600	13.270	134.509	**1.403** (9.46×)	136.319	1.595 (8.32×)	136.945	1.505 (8.82×)

The length-aware reward encourages more efficient computation allocation across streaming and deep stages. Under GRPO, it reduces generation length but slightly hurts accuracy on GSM-symbolic P2 and P1, indicating that flat sequence-level optimization applies length pressure too coarsely. In contrast, HRPO benefits more consistently: compared with *Acc + format*, adding the length reward improves accuracy while reducing total length by 15.2% on P2 and 12.9% on P1, mainly by suppressing redundant streaming-stage thoughts while preserving deep-stage deliberation.

These results show that adaptive rewards and hierarchical advantage assignment are complementary: format rewards make streaming trajectories learnable, length-aware rewards make them efficient, and HRPO assigns these signals to the appropriate reasoning stages.

Performance on Out-of-Domain Cases Table 4 evaluates Qwen3-4B and Qwen3-8B on GSM-Infinite (within-task distribution shift) and LogicNLI (cross-task generalization). HRPO achieves the highest accuracy in all four settings while consistently using fewer tokens than GRPO. On GSM-Infinite, its accuracy gains over the SFT-based baseline increase from 6.7 to 8.7 percentage points when scaling from 4B to 8B, alongside total-length reductions of 2.8% and 10.0%, respectively.

On LogicNLI, GRPO underperforms the SFT-based baseline at both scales, whereas HRPO improves accuracy by 1.5 and 3.5 percentage points at 4B and 8B, respectively. Although 4B HRPO uses more tokens than SFT, 8B HRPO achieves both the highest accuracy and shortest generations on both OOD tasks. These results suggest that hierarchical credit assignment supports transfer across distributions and tasks, with stronger efficiency gains at larger scale.

Analysis of Streaming Latency Table 5 reports the practical latency under Transformers and vLLM backends on the GSM-Symbolic dataset.[7] Compared with read-then-think, streaming reasoning greatly reduces exposed latency, since intermediate reasoning can be overlapped with the arrival of input segments and only the final deep reasoning remains on the critical path. As a result, all streaming methods achieve over $8\times$ latency reduction under both backends. Although AdaSR adaptively reallocates computation between streaming and deep stages, it introduces only minor latency fluctuations compared with StreamingThinker, especially under vLLM. With the vLLM backend, streaming inference is further accelerated, achieving a $4.3\times$ speedup in AdaSR-HRPO latency, which demonstrates that AdaSR can be efficiently deployed in realistic serving scenarios.

5 Conclusion

We presented AdaSR, a reinforcement learning framework that learns when to think and how to allocate computation between reasoning during input reception and final deliberation. Its core algorithm, Hierarchical Relative Policy Optimization (HRPO), aligns advantage assignment with this two-stage structure, combining phase-specific format and length signals with trajectory-level accuracy and efficiency signals. This design couples protocol validity and answer correctness with latency-aware computation allocation. Experiments across model scales and reasoning benchmarks show improved accuracy over supervised streaming reasoning and a better accuracy-efficiency trade-off than GRPO, with stronger joint gains at larger scales. Out-of-domain evaluations further support transfer beyond the training distribution, while latency measurements show post-input delays comparable to the supervised baseline. These findings highlight hierarchical credit assignment as a promising approach to learning efficient reasoning policies for progressively arriving inputs.

[7] Following the StreamingThinker Tong et al. (2025a), we set the input streaming speed to 150 words/min.

AI USE STATEMENT

We used generative AI tools to assist with method implementation and to polish the manuscript. The authors manually reviewed the AI-assisted code to verify implementation correctness and checked the manuscript text for accuracy. The authors take responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe our method and experimental setup in Sections 3 and 4. Implementation details and hyperparameters are provided in Appendix E and Table 8. We further report multiseed results in Table 7 with the latency evaluation protocol specified in Appendix F.

REFERENCES

Victor Agostinelli, Max Wild, Matthew Raffel, Kazi Ahmed Asif Fuad, and Lizhong Chen. Simul-llm: A framework for exploring high-quality simultaneous translation with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1313–1323, 2019.

Siddhant Arora, Haidar Khan, Kai Sun, Xin Luna Dong, Sajal Choudhary, Seungwhan Moon, Xinyuan Zhang, Adithya Sagar, Surya Teja Appini, Kaushik Patnaik, et al. Stream rag: Instant and accurate spoken dialogue systems with streaming tool usage. *arXiv preprint arXiv:2510.02044*, 2025.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18407–18418, 2024.

Joya Chen, Ziyun Zeng, Yiqi Lin, Wei Li, Zejun Ma, and Mike Zheng Shou. Livecc: Learning video llm with streaming speech transcription at scale. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29083–29095, 2025.

Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie Liu, Zhendong Wang, Zhengyuan Yang, Hung-yi Lee, and Lijuan Wang. Shanks: Simultaneous hearing and thinking for spoken language models. *arXiv preprint arXiv:2510.06917*, 2025a.

Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie Liu, Zhendong Wang, Zhengyuan Yang, Hung-yi Lee, and Lijuan Wang. Stitch: Simultaneous thinking and talking with chunked reasoning for spoken language models. *arXiv preprint arXiv:2507.15375*, 2025b.

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

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: A speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. Efficient wait-k models for simultaneous machine translation. In *Proceedings of Interspeech 2020*, 2020.

Qingkai Fang, Shoutao Guo, Yan Zhou, Zhengrui Ma, Shaolei Zhang, and Yang Feng. Llama-omni: Seamless speech interaction with large language models. *arXiv preprint arXiv:2409.06666*, 2024.

In Gim, Seung-seob Lee, and Lin Zhong. Asynchronous llm function calling. *arXiv preprint arXiv:2412.07017*, 2024.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 1053–1062, Valencia, Spain, 2017. Association for Computational Linguistics.

Shoutao Guo, Shaolei Zhang, and Yang Feng. Decoder-only streaming transformer for simultaneous translation. *arXiv preprint arXiv:2406.03878*, 2024a.

Shoutao Guo, Shaolei Zhang, Zhengrui Ma, Min Zhang, and Yang Feng. Agent-simt: Agent-assisted simultaneous machine translation with large language models. *arXiv preprint arXiv:2406.06910*, 2024b.

Shengran Hu and Jeff Clune. Thought cloning: Learning to think while acting by imitating human thinking. In *Advances in Neural Information Processing Systems*, 2023.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2567–2577, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Junming Lin, Zheng Fang, Chi Chen, Zihao Wan, Fuwen Luo, Peng Li, Yang Liu, and Maosong Sun. Streamingbench: Assessing the gap for mllms to achieve streaming video understanding. *arXiv preprint arXiv:2411.03628*, 2024.

Junyan Lin, Junlong Tong, Hao Wu, Jialiang Zhang, Jinming Liu, Xin Jin, and Xiaoyu Shen. Speak while watching: Unleashing true real-time video understanding capability of multimodal large language models. *arXiv preprint arXiv:2601.06843*, 2026.

Jihao Liu, Zhiding Yu, Shiyi Lan, Shihao Wang, Rongyao Fang, Jan Kautz, Hongsheng Li, and Jose M Alvare. Streamchat: Chatting with streaming video. *arXiv preprint arXiv:2412.08646*, 2024.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026a.

Siyuan Liu, Jiahui Xu, Feng Jiang, Kuang Wang, Zefeng Zhao, Chu-Ren Huang, Jinghang Gu, Changqing Yin, and Haizhou Li. Discourse-aware dual-track streaming response for low-latency spoken dialogue systems. *arXiv preprint arXiv:2602.23266*, 2026b.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, et al. Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. *arXiv preprint arXiv:1810.08398*, 2018.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Learning to reason with llms. *OpenAI Blog*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Siqi Ouyang, Oleksii Hrinchuk, Zhehuai Chen, Vitaly Lavrukhin, Jagadeesh Balam, Lei Li, and Boris Ginsburg. Anticipating future with large language model for simultaneous machine translation. *arXiv preprint arXiv:2410.22499*, 2024.

Matthew Raffel, Victor Agostinelli, and Lizhong Chen. Simultaneous masking, not prompting optimization: A paradigm shift in fine-tuning llms for simultaneous translation. *arXiv preprint arXiv:2405.10443*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Li, Mingchuan Zhang, Y.K. Zhang, Yu Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. Diagnosing the first-order logical reasoning ability through LogicNLI. pp. 3738–3747, 2021.

Junlong Tong, Yingqi Fan, Anhao Zhao, Yunpu Ma, and Xiaoyu Shen. Streamingthinker: Large language models can think while reading. *arXiv preprint arXiv:2510.17238*, 2025a.

Junlong Tong, Jinlan Fu, Zixuan Lin, Yingqi Fan, Anhao Zhao, Hui Su, and Xiaoyu Shen. Llm as effective streaming processor: Bridging streaming-batch mismatches with group position encoding. *arXiv preprint arXiv:2505.16983*, 2025b.

Junlong Tong, Zilong Wang, YuJie Ren, Peiran Yin, Hao Wu, Wei Zhang, and Xiaoyu Shen. From static inference to dynamic interaction: A survey of streaming large language models. *arXiv preprint arXiv:2603.04592*, 2026a.

Junlong Tong, Yao Zhang, Anhao Zhao, Yingqi Fan, Yunpu Ma, and Xiaoyu Shen. Proactivellm: Learning active interaction for streaming large language models. *arXiv preprint arXiv:2606.00523*, 2026b.

Lu Wang, Zhuoran Jin, Yupu Hao, Yubo Chen, Kang Liu, Yulong Ao, and Jun Zhao. Think while watching: Online streaming segment-level memory for multi-turn video reasoning in multimodal large language models. *arXiv preprint arXiv:2603.11896*, 2026.

Zhenhailong Wang, Xuehang Guo, Sofia Stoica, Haiyang Xu, Hongru Wang, Hyeonjeong Ha, Xiusi Chen, Yangyi Chen, Ming Yan, Fei Huang, et al. Perception-aware policy optimization for multimodal reasoning. *arXiv preprint arXiv:2507.06448*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Roy Xie, David Qiu, Deepak Gopinath, Dong Lin, Yanchao Sun, Chong Wang, Saloni Potdar, and Bhuwan Dhingra. Interleaved reasoning for large language models via reinforcement learning. *arXiv preprint arXiv:2505.19640*, 2025a.

Zhifei Xie and Changqiao Wu. Mini-omni: Language models can hear, talk while thinking in streaming. *arXiv preprint arXiv:2408.16725*, 2024.

Zhifei Xie, Ziyang Ma, Zihang Liu, Kaiyu Pang, Hongyu Li, Jialin Zhang, Yue Liao, Deheng Ye, Chunyan Miao, and Shuicheng Yan. Mini-omni-reasoner: Token-level thinking-in-speaking in large speech models. *arXiv preprint arXiv:2508.15827*, 2025b.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Haolin Yang, Feilong Tang, Lingxiao Zhao, Xiang An, Ming Hu, Huifa Li, Xinlin Zhuang, Yifan Lu, Xiaofeng Zhang, Abdalla Swikir, et al. Streamagent: Towards anticipatory agents for streaming video understanding. *arXiv preprint arXiv:2508.01875*, 2025b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Wenyi Yu, Siyin Wang, Xiaoyu Yang, Xianzhao Chen, Xiaohai Tian, Jun Zhang, Guangzhi Sun, Lu Lu, Yuxuan Wang, and Chao Zhang. Salmonn-omni: A codec-free llm for full-duplex speech understanding and generation. *arXiv preprint arXiv:2411.18138*, 2024.

Aohan Zeng, Zhengxiao Du, Mingdao Liu, Kedong Wang, Shengmin Jiang, Lei Zhao, Yuxiao Dong, and Jie Tang. Glm-4-voice: Towards intelligent and human-like end-to-end spoken chatbot. *arXiv preprint arXiv:2412.02612*, 2024.

Gengyuan Zhang, Tanveer Hannan, Hermine Kleiner, Beste Aydemir, Xinyu Xie, Jian Lan, Thomas Seidl, Volker Tresp, and Jindong Gu. Avila: Asynchronous vision-language agent for streaming multimodal data interaction. *arXiv preprint arXiv:2506.18472*, 2025a.

Jialiang Zhang, Junlong Tong, Junyan Lin, Hao Wu, Yirong Sun, Yunpu Ma, and Xiaoyu Shen. Think-as-you-see: Streaming chain-of-thought reasoning for large vision-language models. *arXiv preprint arXiv:2603.02872*, 2026.

Qinglin Zhang, Luyao Cheng, Chong Deng, Qian Chen, Wen Wang, Siqi Zheng, Jiaqing Liu, Hai Yu, Chaohong Tan, Zhihao Du, and Shiliang Zhang. Omniflatten: An end-to-end gpt model for seamless voice conversation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025b.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity? *arXiv preprint arXiv:2502.05252*, 2025.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Related Work

Streaming LLMs Streaming LLMs Tong et al. (2026a) aim to move language-model inference from static full-context processing toward dynamic interaction, where input perception, reasoning, output generation, and external actions may overlap. From an application perspective, existing work has explored several forms of real-time interaction. Simultaneous translation studies *read while outputting*, where models learn or impose read/write policies such as wait-k and efficient interaction decision to trade source completeness for latency (Ma et al., 2018; Arivazhagan et al., 2019; Elbayad et al., 2020; Tong et al., 2025b; 2026b), with recent LLM-based and speech-to-speech systems extending this setting to real-time translation (Agostinelli et al., 2024; Raffel et al., 2024; Guo et al., 2024a;b; Ouyang et al., 2024). Streaming video and multimodal systems study *think/speak while watching*, processing frames incrementally, chatting over evolving video streams, or generating reasoning synchronized with incoming visual evidence (Chen et al., 2024; Liu et al., 2024; Chen et al., 2025; Lin et al., 2026; Zhang et al., 2026; Wang et al., 2026). Spoken dialogue systems study *think while hearing* and *speak while thinking*, enabling low-latency or full-duplex speech interaction and, in some cases, hidden reasoning during listening or speaking (Xie & Wu, 2024; Défossez et al., 2024; Fang et al., 2024; Zeng et al., 2024; Zhang et al., 2025b; Yu et al., 2024; Chiang et al., 2025b; Xie et al., 2025b; Chiang et al., 2025a; Liu et al., 2026b). Agentic systems further study *think while acting*, where models interleave reasoning with tool use, retrieval, function calls, or anticipatory actions under streaming observations (Yao et al., 2023; Hu & Clune, 2023; Gim et al., 2024; Arora et al., 2025; Zhang et al., 2025a; Yang et al., 2025b; Xie et al., 2025a). These scenarios show that streaming LLMs are motivated not only by faster token emission, but also by the need to coordinate computation with continuously evolving inputs.

Among these scenarios, *think while reading* provides a clean abstraction for streaming reasoning, where the model processes input segments sequentially, reasons under partial observations, and performs final deliberation after the stream ends. StreamingThinker (Tong et al., 2025a) realizes this paradigm with sentence-level streaming CoT and streaming-specific attention, position encoding, and KV-cache designs, showing how LLMs can reason under streaming visibility constraints. Yet its intermediate reasoning behavior is largely shaped by constructed supervised traces. AdaSR instead studies whether models can learn their own computation policy: when to think, when to skip, and how to allocate computation between online thoughts and final deep reasoning.

Policy Optimization in RLVR Reinforcement Learning with Verifiable Rewards (RLVR) has recently become a central paradigm for improving the reasoning ability of large language models, where policies are optimized using automatically checkable outcome rewards rather than learned reward models (Cobbe et al., 2021; Shao et al., 2024; DeepSeek-AI, 2025). Early RLHF and RLVR systems commonly build upon PPO-style clipped policy objectives (Schulman et al., 2017; Ziegler et al., 2019; Ouyang et al., 2022), while GRPO replaces the critic with group-relative reward normalization, estimating the baseline from multiple rollouts of the same prompt and thereby reducing the cost of value modeling (Shao et al., 2024). Recent studies further refine this paradigm from different perspectives. REINFORCE-style methods revisit critic-free optimization for language-model alignment (Ahmadian et al., 2024). Dr. GRPO analyzes the optimization bias of GRPO and shows that length-dependent loss aggregation can artificially encourage longer responses, especially for incorrect trajectories; it therefore modifies the aggregation and normalization scheme to improve token efficiency while preserving reasoning performance (Liu et al., 2025). DAPO improves the scalability and stability of long-CoT RL by introducing decoupled clipping and dynamic sampling, together with practical techniques such as Clip-Higher, token-level policy-gradient loss, and overlong reward shaping (Yu et al., 2025). GDPO extends GRPO to multi-reward RLVR by decoupling the normalization of different reward components before aggregation, alleviating advantage collapse caused by directly normalizing summed rewards (Liu et al., 2026a). Beyond text-only reasoning, PAPO incorporates perception-aware optimization into RLVR by adding an implicit perception loss and entropy regularization, enabling GRPO- or DAPO-style training to better support multimodal reasoning (Wang et al., 2025). Overall, these methods improve RLVR along dimensions such as critic-free optimization, length-bias correction, training stability, multi-reward composition, and multimodal grounding. However, they still largely optimize completed trajectories at the sequence or token level, without explicitly modeling the hierarchical temporal structure of streaming reasoning. In contrast, our HRPO targets the phase-structured credit assignment problem, assigning advantages

over streaming, deep-reasoning, and global token ranges to better optimize adaptive reasoning under partial and evolving inputs.

B HRPO Analysis

GRPO Limitation for Streaming Reasoning We further formalize why the standard GRPO objective in Eq. 2 is mismatched with streaming reasoning. For a sampled streaming trajectory $o_i = [R_1^{(i)}, \ldots, R_T^{(i)}, R^{(i)}]$, the streaming and deep phases contain L_s^i and $L_d^i = |o_i| - L_s^i$ tokens, respectively, with the token ranges defined in Eq. 4. GRPO first collapses the whole trajectory into a scalar reward R_i and then assigns the resulting group-normalized advantage $\hat{A}_i$ to every token:

$$\nabla_\theta J_{\text{GRPO}}(\theta) \propto \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} m_{i,t} r_{i,t} \hat{A}_i \nabla_\theta \log \pi_\theta(o_{i,t}|q, o_{i,<t})\right], \tag{9}$$

where $m_{i,t}$ denotes whether the clipped surrogate is active. Thus tokens before and after L_s^i receive the same learning signal even though they play different roles: streaming tokens should be judged by the quality and timeliness of partial-observation reasoning, while deep tokens should be judged by full-context integration. **This flat credit assignment creates two failure modes.** If an incorrect or unhelpful streaming segment is rescued by deep deliberation, GRPO still reinforces the streaming tokens because the final reward is high; conversely, useful streaming thoughts are penalized whenever the final answer fails for reasons in the deep phase. Consequently, GRPO cannot tell whether performance changes come from online reasoning, final deliberation, or their interaction, motivating the hierarchical advantage decomposition used by HRPO.

Advantage Decomposition and Policy Gradient We keep the notation in Eq. 5. Starting from the original HRPO objective, the local and global surrogate terms can be grouped by token time step. Streaming tokens receive the streaming local signal and the global signal; deep tokens receive the deep local signal and the same global signal:

$$\begin{aligned}
J_{\text{HRPO}}(\theta) = \mathbb{E}_{\substack{(q,a)\sim\mathcal{D} \\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}}} \Bigg\{ \frac{1}{G}\sum_{i=1}^{G}\Bigg[\lambda \underbrace{\Bigg(\frac{1}{L_s^i}\sum_{t=1}^{L_s^i} \mathcal{C}\big(r_{i,t}^s, \hat{A}_{i,t}^s\big) + \frac{1}{L_d^i}\sum_{t=L_s^i+1}^{|o_i|} \mathcal{C}\big(r_{i,t}^d, \hat{A}_{i,t}^d\big) \Bigg)}_{\text{local clipped surrogate objective}} \\
+ \underbrace{\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\mathcal{C}\big(r_{i,t}^g, \hat{A}_{i,t}^g\big) - \beta\,\mathrm{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})\Big)}_{\text{global clipped surrogate objective}} \Bigg] \Bigg\} \\
= \mathbb{E}_{\substack{(q,a)\sim\mathcal{D} \\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}}} \Bigg\{ \frac{1}{G}\sum_{i=1}^{G}\Bigg[\sum_{t=1}^{L_s^i}\Bigg(\frac{\lambda}{L_s^i}\mathcal{C}\big(r_{i,t}^s, \hat{A}_{i,t}^s\big) + \frac{1}{|o_i|}\mathcal{C}\big(r_{i,t}^g, \hat{A}_{i,t}^g\big) \\
- \frac{\beta}{|o_i|}\mathrm{KL}(\pi_\theta(\cdot|q, o_{i,<t}) \,\|\, \pi_{\text{ref}}(\cdot|q, o_{i,<t}))\Bigg) \\
+ \sum_{t=L_s^i+1}^{|o_i|}\Bigg(\frac{\lambda}{L_d^i}\mathcal{C}\big(r_{i,t}^d, \hat{A}_{i,t}^d\big) + \frac{1}{|o_i|}\mathcal{C}\big(r_{i,t}^g, \hat{A}_{i,t}^g\big) \\
- \frac{\beta}{|o_i|}\mathrm{KL}(\pi_\theta(\cdot|q, o_{i,<t}) \,\|\, \pi_{\text{ref}}(\cdot|q, o_{i,<t}))\Bigg)\Bigg]\Bigg\}.
\end{aligned} \tag{10}$$

This time-wise form makes the effective credit assignment explicit: before the boundary L_s^i, each token is optimized with a streaming-local coefficient λ/L_s^i and a global coefficient $1/|o_i|$; after the boundary, the streaming-local term is replaced by the deep-local coefficient λ/L_d^i.

We next derive the corresponding policy gradient. By linearity of differentiation and expectation,

$$\nabla_\theta J_{\mathrm{HRPO}}(\theta) = \mathbb{E}_{\substack{(q,a)\sim\mathcal{D} \\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\mathrm{old}}}}} \Bigg\{ \frac{1}{G}\sum_{i=1}^{G} \Bigg[\frac{\lambda}{L_s^i}\sum_{t=1}^{L_s^i} \nabla_\theta \mathcal{C}\big(r_{i,t}^s,\, \hat{A}_{i,t}^s\big) + \frac{\lambda}{L_d^i}\sum_{t=L_s^i+1}^{|o_i|} \nabla_\theta \mathcal{C}\big(r_{i,t}^d,\, \hat{A}_{i,t}^d\big) + \frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \nabla_\theta \mathcal{C}\big(r_{i,t}^g,\, \hat{A}_{i,t}^g\big) - \frac{\beta}{|o_i|}\sum_{t=1}^{|o_i|} \nabla_\theta \mathrm{KL}(\pi_\theta(\cdot|q,o_{i,<t}) \,\|\, \pi_{\mathrm{ref}}(\cdot|q,o_{i,<t})) \Bigg] \Bigg\}. \tag{11}$$

For each $\ell \in \{s,d,g\}$, the sampled advantage $\hat{A}_{i,t}^\ell$ and the old-policy denominator in $r_{i,t}^\ell$ are fixed with respect to θ, so

$$\nabla_\theta r_{i,t}^\ell = r_{i,t}^\ell \nabla_\theta \log \pi_\theta(o_{i,t}|q,o_{i,<t}). \tag{12}$$

Therefore the clipped surrogate contributes a policy-gradient term only when the unclipped PPO branch is selected:

$$\nabla_\theta \mathcal{C}\big(r_{i,t}^\ell,\, \hat{A}_{i,t}^\ell\big) = \begin{cases} r_{i,t}^\ell \hat{A}_{i,t}^\ell \nabla_\theta \log \pi_\theta(o_{i,t}|q,o_{i,<t}), & \hat{A}_{i,t}^\ell \geq 0 \text{ and } r_{i,t}^\ell \leq 1+\varepsilon, \\ & \text{or } \hat{A}_{i,t}^\ell < 0 \text{ and } r_{i,t}^\ell \geq 1-\varepsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Substituting Eq. 13 into Eq. 11 gives the time-wise policy gradient:

$$\begin{aligned} \nabla_\theta J_{\mathrm{HRPO}}(\theta) = \mathbb{E}_{\substack{(q,a)\sim\mathcal{D} \\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\mathrm{old}}}}} \Bigg\{ \frac{1}{G}\sum_{i=1}^{G} \Bigg[& \sum_{t=1}^{L_s^i} \left(\frac{\lambda}{L_s^i} m_{i,t}^s r_{i,t}^s \hat{A}_{i,t}^s + \frac{1}{|o_i|} m_{i,t}^g r_{i,t}^g \hat{A}_{i,t}^g \right) \nabla_\theta \log \pi_\theta(o_{i,t}|q,o_{i,<t}) \\ & - \sum_{t=1}^{L_s^i} \frac{\beta}{|o_i|} \nabla_\theta \mathrm{KL}(\pi_\theta(\cdot|q,o_{i,<t}) \,\|\, \pi_{\mathrm{ref}}(\cdot|q,o_{i,<t})) \\ & + \sum_{t=L_s^i+1}^{|o_i|} \left(\frac{\lambda}{L_d^i} m_{i,t}^d r_{i,t}^d \hat{A}_{i,t}^d + \frac{1}{|o_i|} m_{i,t}^g r_{i,t}^g \hat{A}_{i,t}^g \right) \nabla_\theta \log \pi_\theta(o_{i,t}|q,o_{i,<t}) \\ & - \sum_{t=L_s^i+1}^{|o_i|} \frac{\beta}{|o_i|} \nabla_\theta \mathrm{KL}(\pi_\theta(\cdot|q,o_{i,<t}) \,\|\, \pi_{\mathrm{ref}}(\cdot|q,o_{i,<t})) \Bigg] \Bigg\}, \end{aligned} \tag{14}$$

where $m_{i,t}^\ell$ is the binary condition in the first branch of Eq. 13; it is not a new reward or advantage variable, but only records whether the corresponding clipped surrogate is active. Equivalently, if no clipping is active at token t, the streaming-token and deep-token gradient factors reduce to

$$\left(\frac{\lambda}{L_s^i} r_{i,t}^s \hat{A}_{i,t}^s + \frac{1}{|o_i|} r_{i,t}^g \hat{A}_{i,t}^g \right) \nabla_\theta \log \pi_\theta(o_{i,t}|q,o_{i,<t}) \quad \text{for } 1 \leq t \leq L_s^i, \tag{15}$$

and

$$\left(\frac{\lambda}{L_d^i} r_{i,t}^d \hat{A}_{i,t}^d + \frac{1}{|o_i|} r_{i,t}^g \hat{A}_{i,t}^g \right) \nabla_\theta \log \pi_\theta(o_{i,t}|q,o_{i,<t}) \quad \text{for } L_s^i < t \leq |o_i|, \tag{16}$$

with the KL-gradient term subtracted at every token as shown in Eq. 14. At $\theta = \theta_{\text{old}}$, where the ratios equal one, HRPO uses the effective per-token advantages

$$\frac{\lambda}{L_s^i}\hat{A}_{i,t}^s + \frac{1}{|o_i|}\hat{A}_{i,t}^g \quad \text{and} \quad \frac{\lambda}{L_d^i}\hat{A}_{i,t}^d + \frac{1}{|o_i|}\hat{A}_{i,t}^g, \tag{17}$$

for streaming and deep tokens respectively.

C Reward Analysis for Streaming Reasoning

Adaptive reward hacking An intuitive design for local length reward is to add accuracy and format gates, so that only correct and parseable trajectories receive the local length signal. However, our local length term is actually a *penalty*. Consider $\ell_i^s \leq 0$ and the gated design $R_i^{L_s} = R_i^{\text{acc}} R_{i,\text{fmt}}^s \ell_i^s$. Correct and format-valid samples are penalized when they are too long, while incorrect or malformed samples are multiplied by the gate and therefore receive 0 penalty. The gate thus creates a loophole: failing the gate can be better than being correct but verbose.

Rollout	Accuracy	Format	$R_i^{L_s}$
A: correct but long	1	1	-3
B: incorrect	0	1	0
C: incorrect	0	1	0
D: incorrect	0	1	0

After group normalization, the zero-valued incorrect rollouts can obtain positive local length advantages, while the correct long rollout obtains a negative one. The local branch can therefore reward failed trajectories for avoiding the length cost, creating a direct conflict with the global signal.

This failure may appear to be solvable by carefully tuning the coefficient of the local length reward. In the gated regime, however, weighting the reward before computing the advantage cancels the coefficient during normalization, as shown in Eq. 19. Therefore, local length scores should not be gated by final accuracy, and format validity should be handled by masking or filtering at the advantage-computation level rather than by multiplying the negative penalty itself.

Hyperparameter Sensitivity Analysis We analyze why HRPO applies η after component-wise normalization in Eq. 8. Consider a reward-level alternative at any hierarchical level $\ell \in \{s, d, g\}$:

$$R_i^\ell(\eta) = b^\ell + \eta g^\ell X_i^\ell, \qquad \eta > 0, \tag{18}$$

where b^ℓ is the shared additive gate reward, $g^\ell > 0$ is the shared multiplicative gate scale, and X_i^ℓ is the remaining length or efficiency shaping term. In the active gated regime, b^ℓ and g^ℓ are fixed within the sampled group. The normalized advantage is therefore:

$$\begin{aligned}\hat{A}_i^\ell(\eta) &= \frac{R_i^\ell(\eta) - \mu\big(\{R_j^\ell(\eta)\}_{j=1}^G\big)}{\sigma\big(\{R_j^\ell(\eta)\}_{j=1}^G\big)} \\ &= \frac{\eta g^\ell \big(X_i^\ell - \mu(\{X_j^\ell\}_{j=1}^G)\big)}{\eta g^\ell \sigma(\{X_j^\ell\}_{j=1}^G)} = \frac{X_i^\ell - \mu(\{X_j^\ell\}_{j=1}^G)}{\sigma(\{X_j^\ell\}_{j=1}^G)}.\end{aligned} \tag{19}$$

Thus reward-level weighting makes η ineffective under fixed gates. In contrast, Eq. 8 applies η after normalization, so it remains an explicit multiplier on the composed advantage. If a component has zero variance within the group, that component still produces no relative advantage. The corresponding experiments can be found at Table 6.

Hyperparameter Weighting Experiments We investigate the role of different hyperparameters in both reward composition and advantage weighting, with the results summarized in Table 6. Specifically, we study the effects of the format, length, and accuracy reward weights, as well as the advantage weighting coefficient λ, on GRPO and HRPO using Qwen3-1.7B. Based on these comparisons, we select the configuration with Acc: 1, format: 2, length: 0.05, and $\lambda = 0.05$ as the default setting in our paper, since it provides a relatively robust trade-off among accuracy, formatting reliability, and response length.

Table 6: Hyperparameter settings and evaluation results for Qwen3-1.7B. The first two columns report the coefficients of the accuracy and format rewards, while α is the latency discount, η is the adaptive length-reward weight, and λ controls the local-global balance in HRPO. Rows are grouped by ablation theme, and evaluation reports accuracy and total generation length on GSM-symbolic P1/P2 and MetaMathQA.

Reward coefficients		α	η	λ	Method	GSM-symbolic P2		GSM-symbolic P1		MetaMathQA	
Acc.	Format					Acc	Len.	Acc	Len.	Acc	Len.
Baselines											
/	/	0.5	/	/	SFT	0.642	356.910	0.839	286.132	0.688	270.661
1	/	0.5	/	/	GRPO	0.726	392.908	0.877	301.709	0.805	297.310
Format reward and local-global balance without adaptive length reward											
1	2	0.5	/	0.1	HRPO	0.782	498.728	0.895	377.679	0.842	385.627
1	2	0.5	/	0.5	HRPO	0.774	452.712	0.886	341.994	0.816	371.474
1	2	0.5	/	1	HRPO	0.758	381.932	0.872	314.082	0.802	304.650
1	2	0.5	/	1	HRPO	0.756	377.750	0.869	323.723	0.804	321.838
1	2	0.5	/	0.05	HRPO	0.800	433.080	0.884	331.890	0.817	343.084
Adaptive length-reward weight η at fixed $\lambda = 0.5$											
1	2	0.5	0.05	0.5	HRPO	0.642	442.900	0.818	315.474	0.655	461.854
1	2	0.5	0.1	0.5	HRPO	0.532	322.770	0.669	403.125	0.583	531.450
1	2	0.5	0.5	0.5	HRPO	0.134	610.698	0.195	630.160	0.293	621.358
1	2	0.5	1	0.5	HRPO	0.088	136.512	0.154	164.506	0.320	173.537
GRPO and HRPO comparison with stronger local weighting ($\lambda = 1$)											
1	2	0.5	0.05	/	GRPO	0.054	73.670	0.083	73.890	0.194	89.980
1	2	0.5	0.05	1	HRPO	0.620	316.740	0.770	273.110	0.664	334.560
Local-global objective balance λ with adaptive length reward											
1	2	0.5	0.05	0.01	HRPO	0.776	384.980	0.874	299.130	0.813	307.040
1	2	0.5	0.05	0.05	HRPO	0.750	373.060	0.870	290.530	0.801	293.200
1	2	0.5	0.05	0.1	HRPO	0.760	357.980	0.869	279.850	0.810	272.600
1	2	0.5	0.05	1	HRPO	0.620	316.740	0.770	273.110	0.664	334.560
Default setting and latency-discount sensitivity											
1	2	0.5	0.05	/	GRPO	0.758	384.488	0.85	306.792	0.823	317.997
1	2	0.5	0.05	0.05	HRPO	0.788	370.256	0.871	292.004	0.826	302.602
1	2	0.75	0.05	0.05	HRPO	0.800	353.230	0.887	291.355	0.775	276.256
1	2	0.25	0.05	0.05	HRPO	0.784	418.572	0.894	311.095	0.821	344.982
1	2	1	0.05	0.05	HRPO	0.794	351.022	0.876	291.810	0.810	270.148
Format reward coefficient sensitivity at fixed Acc. coefficient = 1											
1	1	0.5	0.05	0.05	HRPO	0.772	361.840	0.858	284.910	0.807	294.730
1	2	0.5	0.05	0.05	HRPO	0.788	370.256	0.871	292.004	0.826	302.602
1	5	0.5	0.05	0.05	HRPO	0.764	405.620	0.852	321.780	0.798	335.440

D Evaluation Details of AdaSR

Multiseed Results with Standard Deviation To assess the robustness of our results, we conduct experiments under multiple random seeds and report the mean performance together with the corresponding standard deviation. The detailed results are presented in Table 7.

Out-of-Domain Experiment Details For the reported Qwen3-4B results in Table 4, our SFT-base model is trained on the math-based datasets P1, P2, and MetaMathQA. GSM-Infinite is an out-of-domain mathematical benchmark, while LogicNLI is a logic-based benchmark used to evaluate the model's out-of-task generalization ability.

E Training Details of AdaSR

While the main text provides the algorithmic workflow of AdaSR, this appendix further details the concrete streaming rollout and stream-aware logits computation, as illustrated in Figure 3.

Procedure of AdaSR algorithm Algorithm 1 summarizes the overall training procedure of AdaSR. Starting from a streaming SFT policy, AdaSR performs group sampling for each input, generates sentence-level streaming rationales followed by a deep reasoning stage, and then computes reward

Table 7: Main multiseed results on streaming reasoning benchmarks with Qwen3-1.7B and Qwen3-4B.

Model	Benchmark	Metric	Read-then-think	StreamingThinker (SFT) (Tong et al., 2025a)	AdaSR-GRPO (Ours)	AdaSR-HRPO (Ours)
Qwen3-1.7B	GSM-symbolic P1	Acc↑	0.634	0.834	0.863 ± 0.0087	**0.872 ± 0.0041**
		Total Len.↓	1598.699	**283.749**	305.108 ± 2.973	**289.930 ± 1.392**
	GSM-symbolic P2	Acc↑	0.424	0.642	0.765 ± 0.0074	**0.773 ± 0.0135**
		Total Len.↓	1866.474	**361.906**	387.673 ± 3.602	**373.438 ± 2.568**
	MetaMathQA	Acc↑	0.848	0.688	0.812 ± 0.0018	**0.815 ± 0.0140**
		Total Len.↓	1281.774	270.661	318.163 ± 1.751	**296.808 ± 8.091**
Qwen3-4B	GSM-symbolic P1	Acc↑	0.636	0.934	0.928 ± 0.0050	**0.935 ± 0.0104**
		Total Len.↓	1717.155	286.992	268.948 ± 0.727	**257.665 ± 6.439**
	GSM-symbolic P2	Acc↑	0.462	0.816	0.871 ± 0.0055	**0.876 ± 0.0112**
		Total Len.↓	1937.396	357.584	336.332 ± 1.203	**317.438 ± 6.752**
	MetaMathQA	Acc↑	0.828	0.860	0.903 ± 0.0050	**0.909 ± 0.0128**
		Total Len.↓	1379.088	274.198	251.699 ± 1.518	**237.388 ± 0.618**

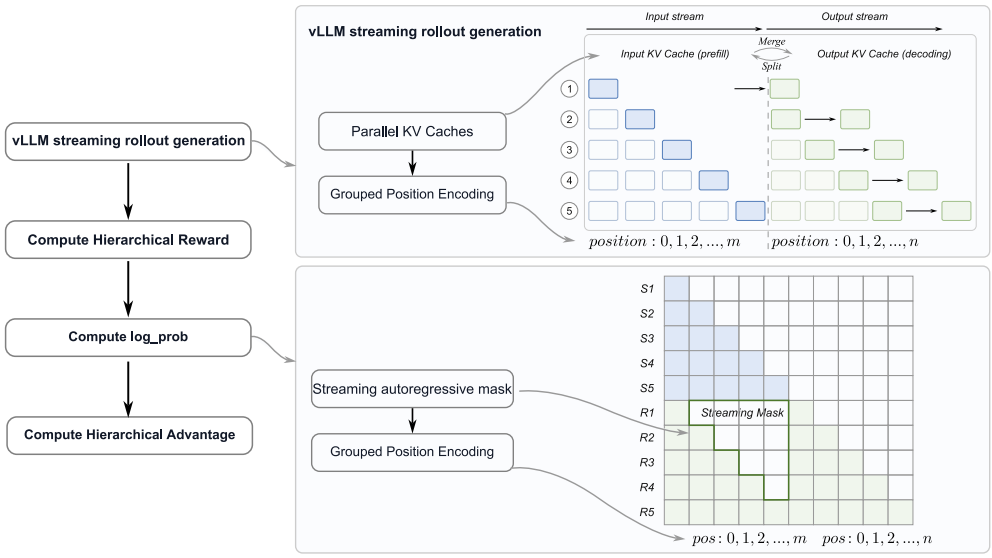

Figure 3: Illustration of the AdaSR training pipeline. The left panel shows the overall training pipeline, where streaming rollouts are generated, hierarchical rewards are computed, log probabilities are evaluated, and hierarchical advantages are assigned. The upper-right panel illustrates the rollout-side implementation, where parallel input/output KV caches and grouped position encoding enable streaming generation with separated prefill and decoding states. The lower-right panel shows the log-probability computation, where a streaming autoregressive mask and grouped position encoding preserve the same partial-observation structure during policy evaluation. Together, these components support consistent streaming rollout generation and streaming-aware hierarchical advantage computation.

components for each sampled trajectory. These rewards are further decomposed into streaming, deep, and global advantages, which are assigned to the corresponding token groups through the hierarchical advantage mechanism. Finally, the policy is optimized with the HRPO objective, allowing the model to improve answer accuracy while explicitly controlling the contribution of different reasoning stages.

Algorithm 1: AdaSR Training with HRPO

Input: Dataset $\mathcal{D}$, policy π_θ initialized from SFT model, group size G, hierarchy weight λ, clip range ε.
Output: Trained policy π_θ.

Copy the initial SFT policy to π_{ref} and freeze it;
for *each training iteration* **do**
  Set $\theta_{\text{old}} \leftarrow \theta$ and hold it fixed for this iteration;
  Sample mini-batch $\mathcal{B} = \{(q_k, a_k)\}$ from $\mathcal{D}$; set $J_{\mathcal{B}}(\theta) \leftarrow 0$;
  for *each* $(q, a) \in \mathcal{B}$ **do**
    Split context into sentences $\{C_1, \ldots, C_T\}$;
    for $i = 1$ **to** G **do** `// Group sampling`
      for $t = 1$ **to** T **do** `// Streaming rollout`
        Prefill C_t with streaming position encoding and attention mask (See Appendix E);
        Decode $R_t^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid Q, C_{\le t}, R_{<t}^{(i)})$ until <EOT>;
      end
      Decode deep reasoning $R^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid Q, C_{\le T}, R_{\le T}^{(i)})$;
      Collect $o_i = [R_1^{(i)}, \ldots, R_T^{(i)}, R^{(i)}]$ with log-probs;
    end
    Compute reward components for each o_i;
    Compose A_i^s, A_i^d, and A_i^g via Eq. 8;
    Assign hierarchical token advantages via Eq. 4;
    Estimate $\widehat{J}_q(\theta)$ via Eq. 5, using the time-grouped form in Eq. 10;
    $J_{\mathcal{B}}(\theta) \leftarrow J_{\mathcal{B}}(\theta) + \widehat{J}_q(\theta)/|\mathcal{B}|$;
  end
  Update θ via gradient ascent on $J_{\mathcal{B}}(\theta)$;
end

Hyperparameters We initialize AdaSR from the corresponding StreamingThinker SFT checkpoint and optimize it with the streaming RL stack described below. Unless otherwise specified, FSDP is used for actor training and reference-policy evaluation, while rollout generation is served by vLLM (Kwon et al., 2023). The reference policy is a frozen copy of the initial SFT policy; the behavior policy is refreshed from the actor before each training iteration and held fixed for that iteration's rollouts and updates. In our main implementation, vLLM rollout uses tensor parallelism over four GPUs. All experiments are conducted on A100 GPUs. The detailed parameters are shown in the Table 8.

Table 8: Hyperparameters in streaming RL training.

Optimization and rewards		**Rollout and generation**	
Parameter	**Value**	**Parameter**	**Value**
Local clipped surrogate weight (λ)	0.05	Max prompt length	8192 tokens
Local length advantage weight (η)	0.05	Max response length	2048 tokens
Latency discount coefficient α	0.5	Max model length	4096 tokens
Scale coefficient τ	Avg. token length	Sampling temperature	0.6
Train batch size	16	Top-p sampling	0.95
PPO mini batch size	16	Top-k sampling	20
PPO micro batch size per GPU	2	Samples per prompt	12
Log-prob micro batch size per GPU	8	Intermediate max tokens	512 tokens
KL coefficient β	0	Compression ratio	0.35
Learning rate	2e-6	Minimum reasoning tokens	20 tokens
Weight decay	0.01	Maximum reasoning tokens	2048 tokens
Warmup ratio	0.05		

veRL Adaptation AdaSR is implemented by adapting the standard verl training loop (Sheng et al., 2024). The original verl pipeline assumes that rollout prompts are fully available before generation and that actor log-probability computation uses a standard full-context causal mask. This assumption is incompatible with streaming reasoning, where each streaming segment must be generated and scored under a partial-observation constraint. We therefore replace the default trainer, rollout engine, and actor forward path at runtime: `RayPPOTrainer` is replaced by a streaming-aware trainer, `vLLMRollout` by `StreamingThinkervLLMRollout`, and the actor model by a Qwen3 forward implementation that accepts streaming metadata. This keeps verl's distributed PPO/GRPO infrastructure, FSDP actor updates, reference policy evaluation, logging, and checkpointing intact, while changing only the parts that define rollout semantics and token visibility.

The training order is important. AdaSR first computes rewards on the raw rollout batch, because reward functions need the generated text and the round-level streaming/deep segment boundaries. Only after reward computation do we repack the rollout into the StreamingThinker training layout. This repacking step reconstructs the source and target segment lists, preserves the generated round boundaries, and writes metadata such as `_lengths`, `source_seg_len`, `target_seg_len`, and `target_seg_roles`. The repacked batch is then used for old log-probability, reference log-probability, advantage assignment, and actor update, ensuring that all policy-gradient quantities are computed under the same streaming visibility.

Rollout Generation The streaming rollout extends vLLM generation from a single full-prompt call to a round-based state machine. For each prompt, the rollout state initially contains only the first revealed source segment and the assistant prefix. At round t, the state is converted to a `TokensPrompt` containing all source segments revealed so far and all previously generated target tokens. The model then decodes a streaming reasoning segment R_t until a streaming stop token such as `<EOT>` is reached. If unrevealed source segments remain, the next sentence is appended to the source side and the request is requeued; otherwise the final round uses the deep-reasoning sampling configuration and decodes until the final termination token.

To preserve StreamingThinker's position semantics during vLLM rollout, each prompt token is annotated with a source or target role, including its segment index. The custom vLLM model-input builder maps these roles to group position IDs: source tokens and target tokens maintain independent position counters that both start from zero. For strict streaming masks, the same segmented roles are also used to build a block-diagonal visibility mask during prefill. Thus, even though vLLM schedules batched generation efficiently, the model observes only the source segments that should be visible at the current streaming step. This implements the StreamingThinker attention and position design inside the RL rollout engine.

Forward Logits Calculation PPO-style updates require the log-probabilities of the sampled response under the old policy, the reference policy, and the updated actor. A subtle issue is that the physical rollout tensor layout differs from the logical streaming layout: prompts are left-padded before rollout, responses are right-padded after generation, and the target sequence contains both streaming and deep reasoning segments. Before each actor forward pass, AdaSR therefore removes invalid padding, concatenates the real source tokens with the real response tokens, and right-pads the resulting logical sequences within the micro-batch. The streaming metadata records the source length, target length, and per-segment boundaries of each sample.

Given this packed representation, the Qwen3 streaming forward pass constructs a sentence-level streaming causal mask. Streaming tokens are allowed to attend to past target tokens and to source segments that have already been revealed, but not to future source segments. The final deep-reasoning segment is allowed to attend to the full source context and previous streaming thoughts. Logits for target tokens are sliced from the positions immediately preceding each response token, and the resulting log-probabilities are written back into the fixed response-width tensors expected by verl. Consequently, the PPO ratio compares old and new policy probabilities for exactly the same sampled tokens under the same streaming attention constraint, while HRPO can assign streaming, deep, and global advantages to their corresponding token ranges.

F Latency Analysis of Streaming Reasoning

Time Delay Time delay measures the wall-clock latency between the arrival of the final input token and the generation of the first answer token, directly reflecting the user-perceived waiting time after the input stream ends. As shown in Figure 4, read-then-think defers all reasoning until the full input is observed, so the entire reasoning process contributes to post-input latency. In contrast, streaming thinking performs intermediate reasoning during input reception, amortizing computation over the input stream and reducing residual latency after the final input segment arrives. Following StreamingThinker Tong et al. (2025a), we set the input reading speed to 150 words per minute.

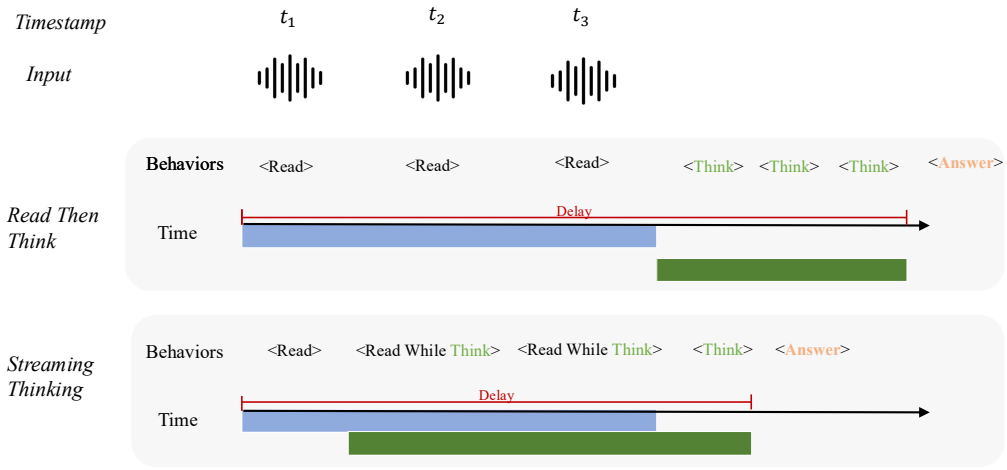

Figure 4: Comparison of reasoning paradigms. Streaming thinking performs reasoning concurrently with incremental input reading, enabling earlier responses and reducing delay. In contrast, the read-then-think paradigm postpones reasoning until all inputs are received, resulting in a larger delay.

G Case Study

Comparison of SFT and RL for Streaming Paradigm for Math Reasoning Figure 5 is an extracted example comparing SFT and RL in mathematical reasoning. As shown in the highlighted section, the RL model performs the calculation correctly in the third segment of the streaming thinking process, whereas the SFT model makes an arithmetic error at the same step. This error in the SFT reasoning is then propagated into the deep thinking stage, ultimately leading to an incorrect answer.

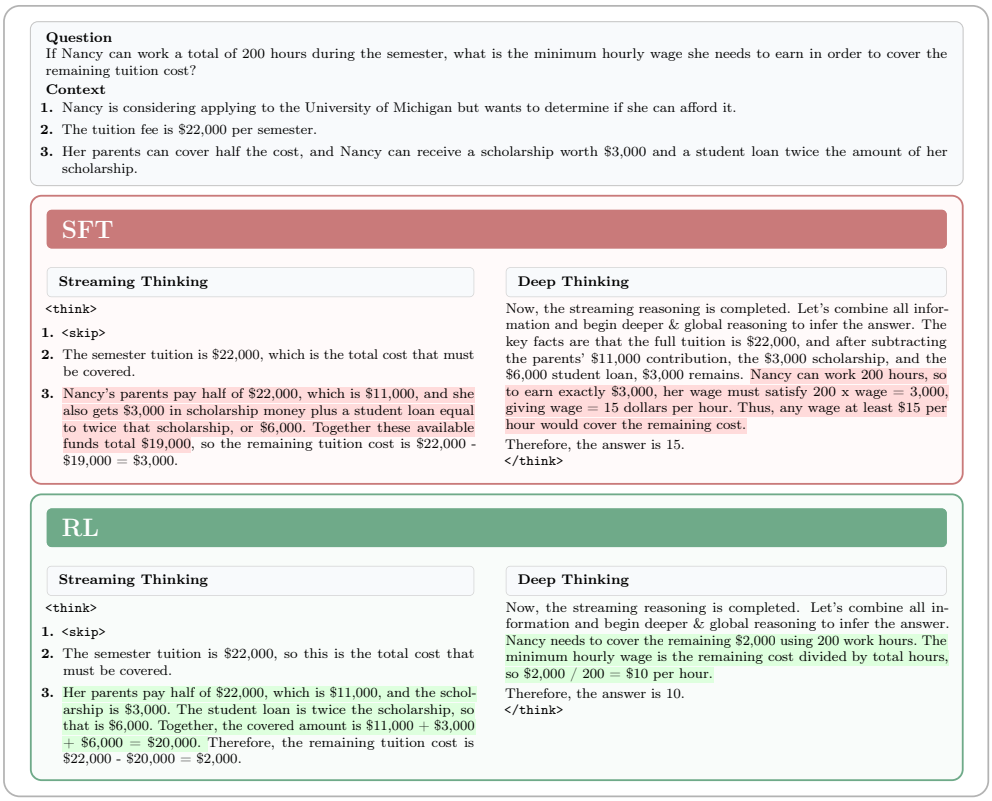

Figure 5: Comparison between SFT and RL on a mathematical reasoning example.