# HYBRIDAGENT: THINK IN LATENT, ACT IN TEXT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

How can we build an agent that thinks beyond human language, yet still acts through it in the external environment? Modern LLM agents still reason and interact through text, while latent reasoning offers an efficient internal medium but lacks native tool access and reasoning transparency. We bridge these two directions with *HybridAgent*, which dynamically navigates between latent reasoning and text-based interaction throughout a unified generation process. Architecturally, a lightweight HybridLink enables seamless transitions between text and latent modes, while the agent's continuous working memory preserves lossless hybrid context across multiple turns. These designs preserve tool-integrated agentic capabilities while making latent thoughts directly inspectable. Building on this architecture, we further develop dedicated post-training algorithms for hybrid reasoning: HybridSFT first stabilizes tool-integrated hybrid generation by teaching the backbone to coordinate with HybridLink, while HybridRL further scales reasoning by learning when to think in latent space and when to act through text. Empirically, we evaluate HybridAgent at 3 model scales across 11 benchmarks covering mathematics, science, medicine, search, and agentic reasoning. Compared with frontier LLM Agents and agentic frameworks spanning both text-based and latent methods, HybridAgent consistently improves performance by 10.4%, while delivering 35.5%–54.7% reduced token overhead, 1.8×–2.6× faster inference, and 2.7×–3.7× cost savings.

## 1 INTRODUCTION

Frontier intelligence no longer scales through the model's reasoning alone, but depends increasingly on its ability to interact with the external world (Team, 2026; Kwa et al., 2026; Sun et al., 2026). To solve a hard problem, an LLM agent must reach beyond the model itself by continually invoking specialized tools, observing feedback from the environment, and adapting its next move (Qu et al., 2025; Silver & Sutton, 2025; Feng et al., 2026). Beneath this expanding repertoire of agent harnesses lies a single lingua franca: *text*, the common medium of both the agent's autoregressive generation and its environment interaction (Sumers et al., 2024; Zhang et al., 2026b). All intermediate reasoning trajectories are generated token by token as text before the agent can act on the environment.

A separate line of work on latent reasoning points in a different direction (Zhu et al., 2025b; Chen et al., 2025). By unfolding computation directly in continuous hidden space, latent models bypass the need to decode and verbalize every intermediate step, providing more efficient internal reasoning within the model's "own language" (Hao et al., 2024; Wang et al., 2025; Yu et al., 2026). However, today's latent reasoners remain largely sealed behind the model boundary. Unlike decoded text, continuous latent vectors (i) lack native pathways for tool invocation across multi-turn agentic reasoning (Chang et al., 2026), and (ii) do not readily expose human-interpretable semantics, leaving them difficult to supervise during training (Chen et al., 2026) and to audit or debug at inference (Korbak et al., 2025). These constraints keep latent reasoning models from operating as capable agents in their own right.

Taken together, agentic interaction and efficient inner reasoning have so far required a different medium each, leaving the two lines of work apart in the development of next-generation agents. We believe this division can be closed by building a new type of agent that combines both strengths to *think beyond text while still acting through it*. We call this unified design **HybridAgent**, one that dynamically navigates between latent reasoning and text-based interaction within a single generation.

Building such an agent is hard, since it needs to handle both dense continuous vectors and sparse discrete tokens as hybrid inputs, and determine when to transition between text and latent modes to balance agentic actions and reasoning efficiency. To offset the hardness, we first revisit the model



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.

architecture and extend its autoregressive generation interface to support hybrid inputs. At the heart of this design is **HybridLink**, a lightweight residual adapter that bridges latent thoughts from the last-layer hidden space back into the input embedding space and enables recurrent computation within latent mode. Then, we realize multi-turn hybrid generation through the agent's own working memory, carrying complete hybrid trajectories of text, latent thoughts, and tool observations across turns. Surprisingly, we find that HybridAgent can render its thoughts in latent mode as human-readable pseudo-text, making latent reasoning transparent and directly inspectable rather than opaque.

To further unlock the potential of hybrid reasoning, we equip HybridAgent with dedicated post-training algorithms. **HybridSFT** first stabilizes tool-integrated hybrid generation through self-conditioned learning, teaching the backbone to coordinate reliably with HybridLink across text and latent modes. Then, **HybridRL** scales agentic reasoning capabilities by training the agent to adaptively balance latent reasoning and text-based interaction. To realize this, we enable policy optimization over hybrid trajectories through hybrid policy factorization and exact rollout replay.

We provide theoretical grounding for HybridAgent through complementary analyses of multi-turn context continuation and hybrid policy optimization. From an architectural standpoint, we show that continuous working memory losslessly preserves the entire hybrid context across different turns. From a learning perspective, the hybrid policy factorization and exact latent-context replay in HybridRL provide a well-founded basis for policy updates over complete hybrid trajectories.

Empirically, we evaluate HybridAgent on 11 benchmarks spanning mathematics, science, medicine, knowledge-intensive search, and challenging agentic reasoning. We instantiate and train HybridAgent with Qwen3-Base models (Yang et al., 2025) across 4B, 8B, and 14B scales. Compared with frontier LLM agents and agentic frameworks spanning both text-based and latent reasoning, HybridAgent achieves an average performance improvement of 10.4% and exhibits strong test-time scalability. In terms of efficiency, HybridAgent reduces token usage by 35.5%–54.7%, delivers 1.8×–2.6× inference speedup, and lowers computational cost by 2.7×–3.7×. Additional in-depth analyses of training dynamics, reasoning auditability, and agentic tool-use reliability reveal a clear division of labor between the two reasoning modes: HybridAgent shifts more internal reasoning into semantically rich and interpretable latent thoughts, while reserving text for external inspection and tool interaction.

## 2 PRELIMINARY

**Hybrid Generation in LLMs.** We consider an LLM agent $f_\theta$ built upon a standard Transformer parameterized by $\theta$. To enable hybrid generation, the agent operates autoregressively in two modes: (i) `text`, which performs explicit generation in the token space, and (ii) `latent`, which performs implicit generation in the continuous representation space. Formally, given an input context $x$ with corresponding input embeddings $E = [e_1, \ldots, e_t] \in \mathbb{R}^{t \times d_h}$, the agent computes the last-layer hidden state $h_t$ through a forward pass, and the hybrid input representation for the next step $t+1$ is:

$$e_{t+1} = \begin{cases} \mathrm{Emb}(y_{t+1}), & y_{t+1} \sim \mathrm{Softmax}(W_{\mathrm{out}} h_t), \quad \text{if } \texttt{text} \text{ mode}, \\ h_t, & \text{if } \texttt{latent} \text{ mode}, \end{cases} \tag{1}$$

where $W_{\mathrm{out}}$ is vocabulary projection matrix, $\mathrm{Emb}(\cdot)$ is token embedding function, and $h_t$ is directly reused as the next latent input in standard latent reasoning. The next hidden state is then computed as:

$$h_{t+1} = f_\theta([E_{\le t}; e_{t+1}]). \tag{2}$$

Throughout inference, the hybrid agent dynamically switches between the two generation modes, producing a hybrid trajectory that balances the model's inner reasoning with external interaction.

**Tool-Integrated Agentic Reasoning.** A hybrid agent can leverage its text mode to interact with external environments. Similar to standard LLM agents, the hybrid agent generates a tool call as a sequence of text, invokes the corresponding tool (e.g., Python, Search), and receives the resulting observations. A complete single-turn *hybrid reasoning trajectory* $\tau$ can be represented as:




(3)

where $[\cdot]$ and $\langle\cdot\rangle$ denote text chunks and latent vectors, respectively. The whole trace is then appended as input for the next generation turn, and the process continues until a final text answer is produced.

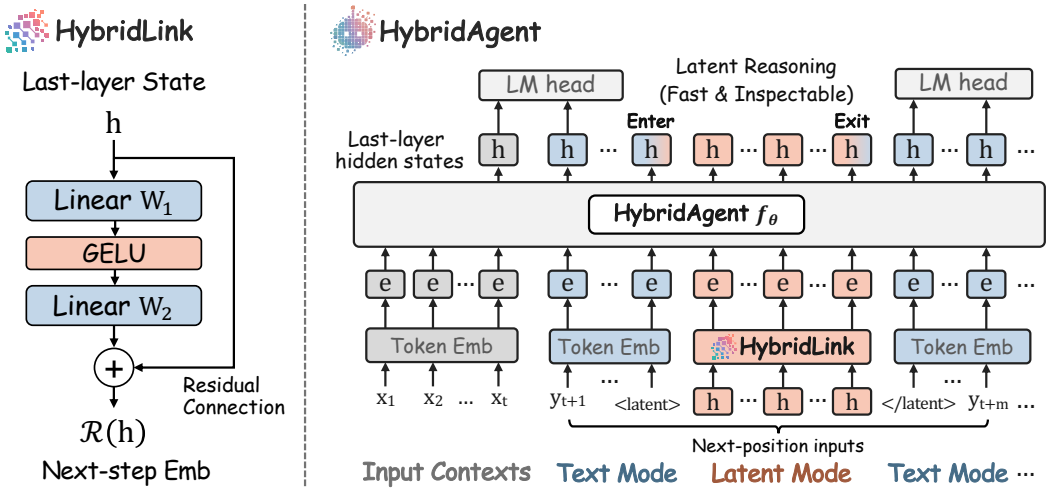


Figure 1: **HybridAgent Architecture Overview.** *Left:* A HybridLink maps last-layer hidden states back into the input embedding space, enabling seamless transitions between text and latent modes. *Right:* The agent performs fast inner thinking while retaining external monitoring and tool interaction.

## 3 Building a Hybrid Agent

We introduce HybridAgent, a hybrid framework that dynamically navigates between explicit text and continuous latent reasoning within a single unified generation process. As illustrated in Figure 1, the architecture preserves an LLM agent's autoregressive generation and tool-calling interfaces while accommodating both text and latent inputs. Below, we first present the hybrid interface for seamless text-latent mode transition (Section 3.1), followed by multi-turn hybrid generation (Section 3.2).

### 3.1 Hybrid Mode Interface

Existing latent reasoning models typically feed the last-layer hidden state $h_t$ directly back as the next-step input $e_{t+1}$. This design is not well suited to hybrid generation, as substituting $h_t$ for a token embedding creates a representation mismatch with the pretrained input space, making transitions between continuous latent states and discrete tokens difficult. To support latent and text modes within a shared input interface, we add a lightweight adapter *HybridLink* to work alongside the backbone.

**HybridLink as a Bridge.** An LLM agent's last-layer hidden states naturally encode the semantics of its current generation, so the HybridLink $\mathcal{R}$ is designed to carry this information from the hidden representation space back into the input embedding space. Formally, given any last-layer hidden state $h_t$, HybridLink simulates its corresponding next-step input embedding as:

$$e_{t+1} = \mathcal{R}(h_t) = h_t + W_2\,\sigma(W_1 h_t)\,, \tag{4}$$

where $W_1, W_2$ are standard linear projections, $\sigma$ is an activation function (e.g., GELU). As the residual connection largely preserves the original semantics of $h_t$, HybridLink only needs to align the distributional difference rather than learn the entire projection from scratch. Accordingly, the link module is lightweight, with approximately $2d_h^2$ parameters (e.g., $\sim$10M parameters for a 4B LLM).

**Dynamic Text-Latent Mode Transition.** Next, we describe how HybridAgent transitions between text and latent modes during generation, together with the entry and exit decisions for each mode.

In text mode, HybridAgent behaves exactly as a standard LLM, sampling the next token from the vocabulary at each step. The process continues until the agent generates one special token `<latent>`, which serves as the entry anchor for the latent mode. For the next forward pass, the agent uses HybridLink for latent reasoning in place of the LM head and token embedding layer.

In latent mode, each step's last-layer hidden state is passed through HybridLink via Eq. 4, allowing HybridAgent to continue its generation flow entirely in latent space without token decoding and re-embedding. This recurrent computation over continuous representations extends the agent's effective reasoning depth, enabling more expressive internal reasoning. The latent reasoning span terminates when the semantics of the last-layer hidden state $h_t$ align with the exit anchor `</latent>`. Specifically, at any latent step $t$, the latent mode closes by satisfying the condition:

$$\arg\max_{v\in\mathcal{V}} \cos\big(\mathcal{R}(h_t),\,\mathrm{Emb}(v)\big) = \texttt{</latent>}. \tag{5}$$

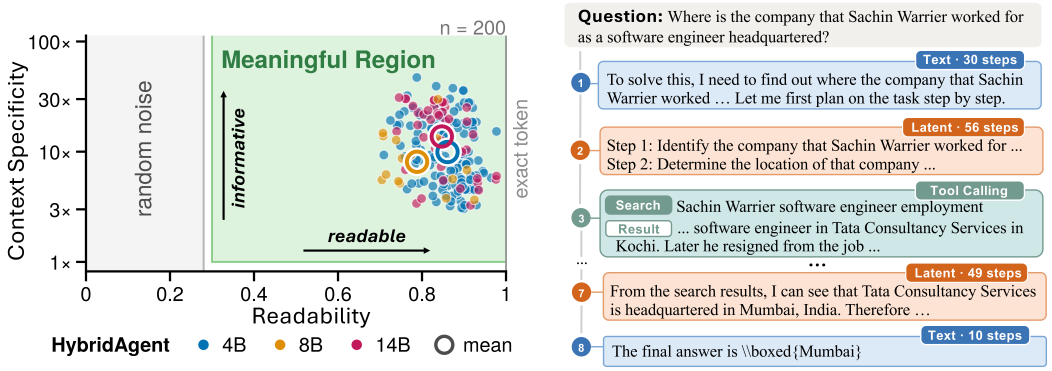


Figure 2: **Interpretable Latent Mode in HybridAgent.** All latent steps can be expressed as readable pseudo-text carrying their semantics for external inspection. *Left:* Quantitatively, latent thoughts are highly readable and context-specific (see Appendix E). *Right:* Qualitatively, latent pseudo-text stays semantically aligned with surrounding text reasoning and tool calls. ("Steps" counts forward passes).

HybridAgent then returns to its text mode, and this transition continues during generation. At each step, the agent adaptively decides whether to remain in its current mode or switch, allowing the length and frequency of text and latent spans to emerge dynamically rather than follow preset configurations.

**Interpretable Latent Mode.** During hybrid generation, a useful property of HybridAgent is that its latent mode remains decoding-free yet directly interpretable. To inspect any latent reasoning span, one can easily recover its semantics by mapping each latent step produced by HybridLink to its nearest token embedding, as in Eq. 5. We illustrate this interpretability property clearly in Figure 2.

### 3.2 Multi-Turn Hybrid Generation over Working Memory

In standard LLM agents, each turn's generated content and tool outputs are appended to the textual context and re-encoded as the next turn's input prefix. Hybrid trajectories, however, interleave text tokens with latent vectors and can't be reconstructed through conventional tokenization alone. Instead, HybridAgent carries the complete context across turns through its continuous working memory.

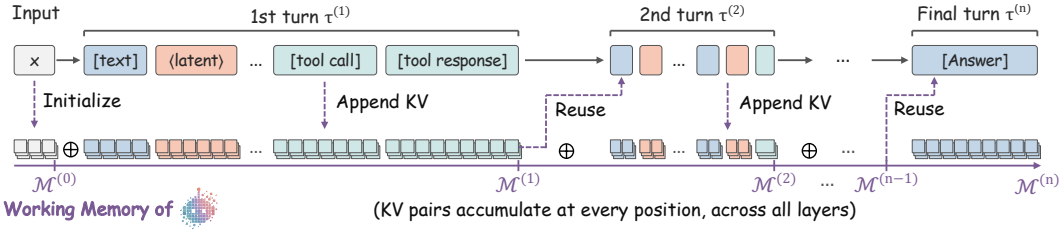


Figure 3: Multi-Turn Hybrid Generation over KV-based Working Memory.

**Cross-Turn Working Memory.** Recall that a single-turn hybrid trajectory $\tau$ in Eq. 3 consists of model-generated text, latent thoughts, and external tool interactions. During input construction, all contents are mapped into the same input space through either the input embedding layer or HybridLink. This shared space allows the Transformer to process all three components uniformly and collect their KV entries throughout generation. We leverage these accumulated KV entries as the HybridAgent's continuous working memory to achieve multi-turn hybrid generation.

As illustrated in Figure 3, HybridAgent first initializes its working memory $\mathcal{M}$ from the input context $x$ and then accumulates the KV entries generated throughout the first-turn trajectory $\tau^{(1)}$, i.e.,

$$\begin{aligned} \mathcal{M}^{(0)} &= \left[K_{\le |x|}, V_{\le |x|}\right], \\ \mathcal{M}^{(1)} &= \mathcal{M}^{(0)} \oplus \left[K_{\le t}, V_{\le t}\right], \quad t = 1, \ldots, |\tau^{(1)}|. \end{aligned} \tag{6}$$

Here, $\mathcal{M}^{(1)}$ preserves the complete first-turn hybrid context in positional order across all Transformer layers. For each subsequent turn, the agent directly resumes generation from the current working memory and updates it by appending newly generated KV entries at each step. This process continues across the sequence of hybrid trajectories $\{\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(n)}\}$ until the final answer is produced. Overall, multi-turn hybrid generation proceeds over one continuous working memory flow, preserving the complete hybrid context across turns without requiring intermediate token-space reconstruction.

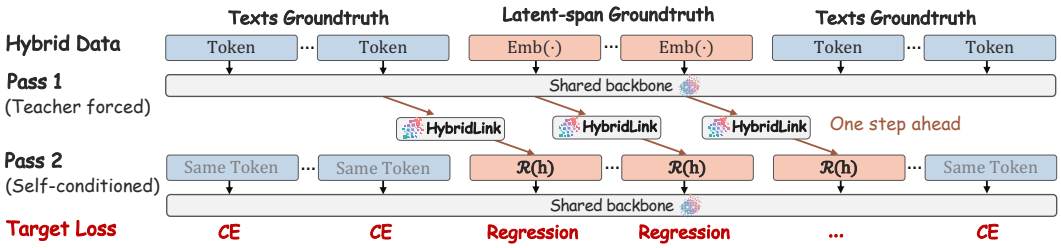


Figure 4: **Self-Conditioned HybridSFT.** A teacher-forced first pass produces latent inputs via HybridLink, replacing ground-truth latent embeddings in a second pass for joint text-latent supervision.

## 4 Learning to Perform Hybrid Reasoning

With the overall framework in place, we now turn to training. As HybridAgent leaves the backbone architecture intact, no pre-training from scratch is required. Instead, we adapt the standard agentic post-training pipeline of supervised fine-tuning and reinforcement learning to the hybrid setting. The two training stages serve complementary roles: SFT teaches the agent to coordinate with HybridLink and instantiate reliable tool use for stable hybrid generation (Section 4.1), while RL further scales reasoning by learning when to think in latent space and when to act through text (Section 4.2).

### 4.1 Stabilize Generation via Hybrid Supervised Fine-tuning

As a first step toward stabilizing hybrid reasoning in HybridAgent, we design the entire HybridSFT stage through a hybrid data curation pipeline and a staged training procedure.

**Hybrid Data Recipe.** To curate high-quality training data, we filter existing text-based agentic SFT data from multiple sources (Yu et al., 2025; Dong et al., 2025b) and convert it into hybrid trajectories through automatic annotation. We assign reasoning steps to latent mode based on both LLM Experts' judgments and target models' own prediction entropy, while retaining a small set of key steps for tool interactions and external debugging/auditing in text mode (see Appendix D for data curation details).

**Format and HybridLink Activation.** As a warm start, we adapt the backbone to the hybrid data format and initialize the HybridLink. For format activation, we perform standard text SFT to teach the backbone when to enter and exit latent mode by emitting the two special anchor tokens. We then freeze the backbone and train HybridLink to align last-layer hidden states at latent positions with their target input embeddings. For a training instance $(x, y) \in \mathcal{D}_{\text{SFT}}$, the activation objectives are:

$$\begin{aligned} \mathcal{L}_{\text{Backbone}} &= \text{CE}\big(f_\theta(x),\, y\big), \\ \mathcal{L}_{\text{HybridLink}} &= \lambda_{\cos}\Big(1 - \cos\big(\mathcal{R}(h_{\text{lat}}),\, \text{Emb}(y_{\text{lat}})\big)\Big) + \lambda_{\text{mse}}\big\|\mathcal{R}(h_{\text{lat}}) - \text{Emb}(y_{\text{lat}})\big\|_2^2, \end{aligned} \tag{7}$$

where $h_{\text{lat}}$ is the last-layer hidden states at latent positions, $y_{\text{lat}}$ is the corresponding ground-truth text tokens, and $\lambda_{\cos}, \lambda_{\text{mse}}$ balance directional and magnitude alignment, respectively.

**Self-Conditioned HybridSFT.** The activation provides a well-aligned initialization, but the backbone has not yet learned to reason over latent thoughts produced by its own HybridLink. We therefore adopt a *two-pass self-conditioned training* scheme (See Figure 4). In the first pass, HybridAgent is teacher-forced on the ground-truth hybrid sequence. HybridLink maps the generated last-layer hidden states at latent positions one step ahead to form latent inputs for the next pass. In the second pass, HybridAgent processes the same sequence but replaces the ground-truth latent embeddings with its own latent inputs. We jointly supervise text and latent positions on the second pass as:

$$\mathcal{L}_{\text{HybridSFT}} = \mathcal{L}_{\text{Backbone}} + \mathcal{L}_{\text{HybridLink}}, \tag{8}$$

where the CE loss $\mathcal{L}_{\text{Backbone}}$ and regression loss $\mathcal{L}_{\text{HybridLink}}$ are applied only at text and latent positions, respectively. Overall, HybridSFT trains the agent to coordinate its backbone with HybridLink for coherent reasoning across text and latent modes, supporting stable hybrid generation at inference.

### 4.2 Scale Reasoning via Hybrid Reinforcement Learning

A HybridAgent cannot straightforwardly adopt popular RL algorithms such as PPO (Schulman et al., 2017) or GRPO (Shao et al., 2024), as they assume every generated position carries a token-level

**Algorithm 1** Hybrid Reinforcement Learning (HybridRL)

**Input:** Agent $f_\theta$ (SFT initialized in Section 4.1), HybridLink $\mathcal{R}$, task set $\mathcal{D}_{\text{RL}}$, group size $G$, $\eta$

**for** each training step **do**
  Sample $x \sim \mathcal{D}_{\text{RL}}$; Collect $\{\tau_i\}_{i=1}^{G} \leftarrow f_\theta(x)$ ▷ Hybrid Policy Rollout
  Record $\tau_i$ with latent $z_i = \mathcal{R}(H_i)$ (Eq. 4), text $y_i$, and tool observations $o_i$
  Compute rewards $r_i$ (Eq. 11) and group-normalized advantages $A_i$
  Replay $\tau_i$ with $(z_i, y_i, o_i)$ to obtain $\pi_\theta(y_{i,t} \mid \tau_{i,<t})$ ▷ Exact Rollout Replay
  Compute $\rho_{i,t}$ via Eq. 9 and Eq. 10 ▷ Hybrid Policy Factorization
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{RL}}$ (standard RL objective) ▷ Hybrid Policy Update
**end for**

likelihood. Specifically, at each position $t$, standard RL computes the per-token importance ratio as:

$$\rho_t = \frac{\pi_\theta\big(y_t \mid y_{<t}\big)}{\pi_{\theta_{\text{old}}}\big(y_t \mid y_{<t}\big)}, \tag{9}$$

where $\pi_\theta, \pi_{\theta_{\text{old}}}$ denote the current and rollout policies. At latent positions, however, HybridAgent generates continuous vectors rather than sampled tokens, leaving no token-level likelihood. Thus, the importance ratio $\rho_t$ is undefined. We address this incompatibility with two simple components.

**Hybrid Policy Factorization.** Recall that, in a hybrid trajectory, latent thoughts are computed entirely in the continuous latent space through HybridLink, and tool responses are the masked observations from the environment. Thus, policy sampling during rollout is confined to model-generated text positions. To justify why policy optimization needs only to operate over these sampled positions, we formalize the *hybrid policy factorization* below with detailed theoretical analyses.

**Proposition 4.1** (Hybrid Policy Factorization)**.** *Let $\tau$ be a hybrid trajectory generated by the policy $\pi_\theta$ (induced by $f_\theta$) with fixed HybridLink $\mathcal{R}$. Then the hybrid policy factorizes as:*

$$\pi_\theta(\tau \mid x) \;=\; \prod_{t \in \mathcal{T}} \pi_\theta\big(y_t \mid x,\, \tau_{<t}\big), \tag{10}$$

*where $\mathcal{T}$ is the set of sampled text positions of trajectory $\tau$.*

The proof and additional theoretical justifications are provided in Appendix B. Proposition 4.1 shows that likelihood-ratio updates over sampled text positions are sufficient to optimize the full hybrid trajectory, making the importance ratio in Eq. 9 well-defined for the HybridAgent policy.

**Policy Update via Exact Rollout Replay.** Policy optimization generally involves rebuilding a differentiable computation graph for the rollouts. Applying this to HybridAgent further requires recomputing sampled text-token likelihoods conditioned on the same preceding latent thoughts used during rollout. We satisfy this requirement via *exact latent-context replay*. During hybrid rollout, we record latent thoughts alongside sampled text tokens and tool observations. During policy updates, we replay the full hybrid trajectory in embedding space with a single differentiable forward pass, keeping the recorded latent inputs unchanged while recomputing the sampled likelihoods in Eq. 10.

Algorithm 1 outlines the full HybridRL training procedure. We keep the HybridLink fixed to preserve deterministic latent transitions. For the reward design, we simply formulate it as:

$$r(\tau) = r_{\text{correct}} + r_{\text{format}} + r_{\text{tool}}, \tag{11}$$

where $r_{\text{correct}}$ measures answer correctness, $r_{\text{format}}$ penalizes invalid formatting, and $r_{\text{tool}}$ encourages multi-tool use. We impose no explicit reward for mode usage, allowing HybridAgent to adaptively balance text and latent generation and discover effective hybrid reasoning strategies during RL.

## 5 Empirical Evaluations

**Tasks and Datasets.** We conduct comprehensive evaluations of HybridAgent on 11 benchmarks across various domains: (i) *Mathematical Reasoning*, including AIME 25 (math ai, 2025), AIME 26 (MathArena, 2026), and MATH500 (HuggingFaceH4, 2023); (ii) *Knowledge-Intensive Reasoning*, including GPQA-Diamond (Rein et al., 2023), MedQA (Yang et al., 2024), and MuSiQue (Trivedi et al., 2022); and (iii) *Agentic Reasoning*, including General AI Assistants (GAIA) (Mialon et al., 2024), Humanity's Last Exam (HLE) (Center for AI Safety et al., 2026), WebWalker (Wu et al., 2025), SealQA (Pham et al., 2026), and DeepSearchQA (DSQA) (Gupta et al., 2026). We adopt the standard evaluation metric for each dataset. Additional benchmark details are in Appendix H.1.

Table 1: **Main Results of HybridAgent over Agentic Reasoning Tasks.** We report GAIA and WebWalker by difficulty level, HLE by category (NS/CE/SF), SealQA on the Seal-0 ("S-0") and Seal-Hard ("Hard") splits, and DSQA by full-correct rate ("Acc.") and F1 score. Across 5 independent runs, the average standard deviations of HybridAgent are $\pm 0.014$ in accuracy, with a corresponding 95% CI of $\pm 0.017$. At 8B and 14B scales, HybridAgent consistently outperforms advanced LLM agents, agentic frameworks, and matched text and latent baselines. Best performance is **highlighted**.

| Method | GAIA | | | | WebWalker | | | | HLE | | | | SealQA | | DSQA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Lv.1 | Lv.2 | Lv.3 | Avg. | Easy | Med. | Hard | Avg. | NS | CE | SF | Avg. | S-0 | Hard | Acc. | F1 |
| **Frontier LLM Agents** | | | | | | | | | | | | | | | | |
| QwQ-32B | 30.9 | 6.5 | 5.2 | 18.9 | 7.5 | 2.1 | 4.6 | 4.3 | 11.5 | 7.3 | 5.2 | 9.6 | 1.8 | 10.6 | 11.3 | 23.9 |
| GLM-5.3-Flash | 38.5 | 28.8 | 8.3 | 30.1 | 8.9 | 8.3 | 9.9 | 9.0 | 13.3 | 14.7 | 16.8 | 14.2 | 9.0 | 22.0 | 16.0 | 38.5 |
| DeepSeekV4.1-Flash | 43.6 | 42.3 | 16.7 | 39.8 | 22.2 | 16.7 | 12.7 | 16.5 | 28.5 | 25.3 | 30.5 | 28.4 | 14.4 | 33.5 | 19.3 | 38.6 |
| Kimi-K3 | 46.2 | 42.3 | 25.0 | 41.7 | 11.1 | 11.9 | 7.0 | 10.0 | 29.4 | 32.0 | 37.9 | 31.4 | 11.7 | 28.0 | 21.7 | 46.0 |
| GPT-5.6-Luna | 48.7 | 32.7 | 25.0 | 37.9 | 15.6 | 8.3 | 11.3 | 11.0 | 30.9 | 25.3 | 25.3 | 29.0 | 11.7 | 29.1 | 23.3 | 49.4 |
| **Agentic Frameworks** | | | | | | | | | | | | | | | | |
| Search-o1 | 46.2 | 23.1 | 0.0 | 29.1 | 11.1 | 21.4 | 16.9 | 17.5 | 6.4 | 4.0 | 10.5 | 6.8 | 3.6 | 11.8 | 12.3 | 25.3 |
| WebThinker | 48.7 | 26.9 | 8.3 | 33.0 | 13.3 | 23.8 | 18.3 | 19.5 | 7.0 | 4.0 | 9.5 | 7.0 | 4.5 | 12.6 | 11.7 | 24.7 |
| ReAct | 48.7 | 25.0 | 8.3 | 32.0 | 11.1 | 20.2 | 12.7 | 15.5 | 5.8 | 5.3 | 10.5 | 6.6 | 7.2 | 15.0 | 14.6 | 32.3 |
| **8B-level Latent & Text Methods** | | | | | | | | | | | | | | | | |
| Qwen3-8B (thinking) | 28.1 | 15.4 | 16.7 | 20.4 | 0.0 | 2.4 | 2.8 | 2.0 | 3.9 | 2.7 | 8.4 | 4.6 | 2.7 | 5.9 | 3.3 | 14.6 |
| Coconut | 7.7 | 1.9 | 0.0 | 3.9 | 0.0 | 1.2 | 0.0 | 0.5 | 1.5 | 0.0 | 1.1 | 1.2 | 0.0 | 1.6 | 1.7 | 6.1 |
| SoftThinking | 23.1 | 9.6 | 8.3 | 14.6 | 0.0 | 1.2 | 1.4 | 1.0 | 3.6 | 2.7 | 6.3 | 4.0 | 0.0 | 3.9 | 2.3 | 11.2 |
| GRPO | 48.7 | 25.0 | 8.3 | 32.0 | 28.9 | 32.1 | 28.2 | 30.0 | 7.9 | 4.0 | 10.5 | 7.8 | 15.3 | 23.6 | 17.3 | 33.8 |
| ARPO | **53.9** | 32.7 | 16.7 | 38.8 | 31.1 | 35.7 | 28.2 | 32.0 | 7.3 | 6.7 | 15.8 | 8.8 | 18.9 | 26.0 | 18.0 | 37.2 |
| **HybridAgent (8B)** | 51.3 | **44.2** | **25.0** | **43.7** | **48.9** | **45.2** | **35.2** | **42.5** | **14.2** | **8.0** | **20.0** | **14.4** | **22.5** | **35.8** | **20.0** | **40.7** |
| **14B-level Latent & Text Methods** | | | | | | | | | | | | | | | | |
| Qwen3-14B (thinking) | 33.3 | 13.5 | 0.0 | 19.4 | 6.7 | 2.4 | 4.2 | 4.0 | 5.5 | 6.7 | 11.6 | 6.8 | 3.6 | 7.9 | 4.0 | 15.1 |
| Coconut | 10.3 | 5.8 | 0.0 | 6.8 | 2.2 | 2.4 | 1.4 | 2.0 | 3.9 | 4.0 | 5.3 | 4.2 | 0.0 | 2.8 | 2.0 | 7.5 |
| SoftThinking | 23.1 | 11.5 | 8.3 | 15.5 | 6.7 | 3.6 | 7.0 | 5.5 | 3.9 | 5.3 | 9.5 | 5.2 | 2.7 | 6.3 | 5.7 | 15.4 |
| GRPO | 51.3 | 34.6 | 0.0 | 36.9 | 35.6 | 42.9 | 35.2 | 38.5 | 7.9 | 6.7 | 12.6 | 8.6 | 17.1 | 24.4 | 18.6 | 43.9 |
| ARPO | **56.4** | 40.4 | 16.7 | 43.7 | 40.0 | 44.1 | 36.6 | 40.5 | 10.3 | 10.7 | 13.7 | 10.0 | 17.1 | 22.8 | 16.3 | 38.2 |
| **HybridAgent (14B)** | 53.8 | **55.8** | **33.3** | **52.4** | **48.9** | **52.4** | **46.5** | **49.5** | **16.1** | **13.3** | **17.9** | **16.0** | **24.3** | **37.8** | **21.7** | **46.8** |

**Baselines.** For baseline comparisons, we evaluate HybridAgent against (i) *Text-based Agentic Methods*, covering standard text SFT (with tools) and multi-turn tool-integrated RL methods such as GRPO (Shao et al., 2024) and ARPO (Dong et al., 2025b); (ii) *Latent Reasoning Methods*, including SoftThinking (Zhang et al., 2025), Coconut (Hao et al., 2024), and SoftCoT (Xu et al., 2025b); (iii) *Frontier LLM Agents*, including GPT-5.6-Luna (OpenAI, 2026), Kimi-K3 (Team, 2026), GLM-5.3-Flash (GLM-5-Team et al., 2026), DeepSeek-V4.1-Flash (DeepSeek-AI, 2026), QwQ-32B (Team, 2025); and (iv) *Agentic Frameworks*, including ReAct (Yao et al., 2023), Search-o1 (Li et al., 2025a), and WebThinker (Li et al., 2025b). Additional baseline details are provided in Appendix H.2.

**Models and Implementation Details.** We instantiate HybridAgent with Qwen3-Base backbones (Yang et al., 2025) at 4B, 8B, and 14B scales. We then apply HybridSFT with 54K curated hybrid trajectories, followed by HybridRL with 12K deep reasoning and 1K agentic search examples. Each RL rollout contains 512 trajectories ($64 \times 8$). HybridAgent and baselines are provided with access to sandboxed Python and live search (e.g., Tavily). We conduct all experiments with five independent seeds. Additional training/inference details and hyperparameter setups are provided in Appendix H.3.

### 5.1 Scaling Performance via Hybrid Reasoning

**Main Results.** Tables 1 and 2 report the overall performance of HybridAgent across 11 benchmarks spanning math, science, medicine, and knowledge search domains. On challenging agentic reasoning benchmarks, HybridAgent delivers competitive performance over larger frontier LLM agents and specialized tool-use frameworks, with HybridAgent (14B) improving over the strongest agentic baseline by an average of 17.4%. In addition, under matched model sizes and training budgets, HybridSFT and HybridRL demonstrate consistent training effectiveness and improve over corresponding text/latent baselines by 18.5% at the SFT stage and 10.4% at the RL stage. Together, HybridAgent combines the strengths of text-based agentic tool use with faster and more expressive latent reasoning, allowing the agent to interact externally while achieving deeper thinking depth internally.

Table 2: **Main Results of HybridAgent on Math and Knowledge-Intensive Search.** We evaluate performance across both SFT and RL stages. Small-size benchmarks like AIME25/26 are reported on avg@16 accuracy. Across all tasks, HybridAgent yields an average std. of ±0.031 (math) and ±0.018 (search), with corresponding 95% CI of ±0.038 and ±0.022. Under the same training budgets and model setups, HybridSFT and HybridRL improve both text-based and latent reasoning methods.

| Method | AIME25 | AIME26 | MATH500 | GPQA-D | MedQA | MuSiQue |
|---|---|---|---|---|---|---|
| **Qwen3-8B-Base** | | | | | | |
| Coconut | 13.1 | 9.6 | 51.4 | 26.8 | 61.7 | 14.0 |
| SoftCoT | 16.5 | 17.9 | 63.6 | 28.9 | 62.0 | 19.5 |
| Standard Text SFT | 30.0 | 28.3 | 78.2 | 32.3 | 69.7 | 23.5 |
| **HybridAgent (SFT)** | 30.8 | 29.2 | 78.4 | 31.8 | 70.0 | 26.5 |
| GRPO | 36.3 | 39.4 | 82.0 | 43.9 | 74.3 | 27.5 |
| ARPO | 37.1 | 41.5 | 82.8 | 46.5 | 74.7 | 28.5 |
| **HybridAgent (RL)** | **39.2** | **43.3** | **84.6** | **48.0** | **75.3** | **32.0** |
| **Qwen3-14B-Base** | | | | | | |
| Coconut | 16.3 | 15.2 | 61.2 | 29.3 | 65.3 | 17.5 |
| SoftCoT | 22.7 | 23.8 | 74.0 | 36.4 | 67.7 | 20.0 |
| Standard Text SFT | 30.8 | 31.7 | 83.8 | 43.9 | 78.0 | 28.0 |
| **HybridAgent (SFT)** | 32.5 | 31.7 | 84.2 | 45.5 | 78.7 | 29.5 |
| GRPO | 40.8 | 42.5 | 87.4 | 52.5 | 81.0 | 32.0 |
| ARPO | 42.1 | 43.3 | 87.0 | 54.0 | 80.3 | 31.5 |
| **HybridAgent (RL)** | **43.3** | **45.8** | **88.2** | **55.1** | **81.7** | **34.5** |

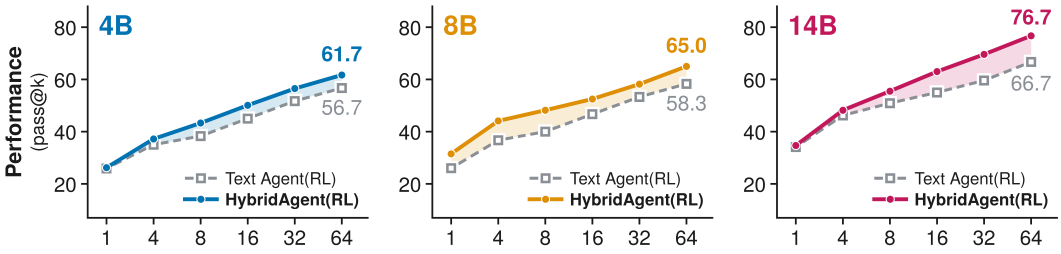


Figure 5: **Scalability of HybridAgent at Test-time.** We evaluate pass@k $(4, 8, 16, 32, 64)$ performance on 100 hard math and science questions and compare with text agents trained with GRPO.

**Scaling Hybrid Reasoning at Test Time.** To further examine the test-time scalability of our hybrid reasoning paradigm, we increase inference compute budgets through additional sampling and evaluate pass@k performance. Figure 5 shows that HybridAgent exhibits a clear upward trend as sampling increases across all model scales, while benefiting more from increased test-time compute than agents with text-only RL. Additional 4B-scale experiments and case studies are in Appendix F and J.

### 5.2 Efficiency Advantage on Hybrid Generation

Beyond performance effectiveness, a key architectural advantage of HybridAgent is that latent mode enables efficient generation without costly explicit token decoding. We evaluate this property empirically along three complementary dimensions: (i) *token throughput*, (ii) *wall-clock inference time*, and (iii) *actual computation cost*. As shown in Figures 6 and 7, HybridAgent adaptively balances its hybrid reasoning across text and latent modes by using 2.7× fewer text tokens, reducing total generation steps by up to 54%, and achieving 2.1× faster inference than text-based baselines. Together with the cost analyses in Table 4 (see Appendix F.2), HybridAgent generates more concise and refined reasoning trajectories, reducing inference overhead and cost.

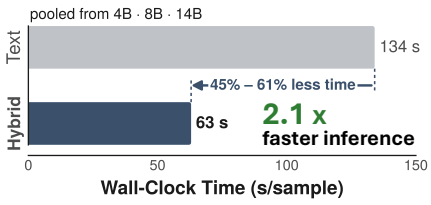


Figure 6: Inference Time Speedup of HybridAgent across Model Scales.

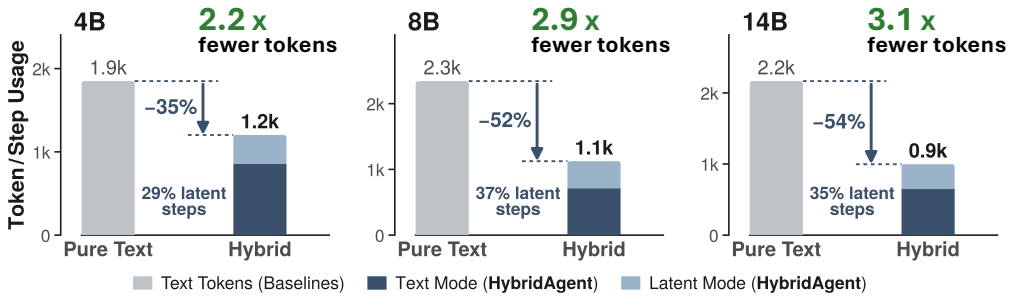


Figure 7: **Efficient Hybrid Generation of HybridAgent across Model Scales.** HybridAgent reduces both text-token usage and total generation steps compared with matched text-based LLM agents.

### 5.3 In-Depth Analyses on HybridAgent

We further analyze HybridAgent from three complementary perspectives to assess the method design and better understand its hybrid reasoning behavior. **(i) Training Dynamics**: We study how self-conditioning in HybridSFT improves reasoning performance (Table 5) and latent interpretability (Figure 9), and how HybridRL shifts more reasoning into latent mode while strengthening tool interaction (Figure 10). **(ii) Latent-Mode**: We intervene on latent content, ordering, and reasoning depth to examine the causal reliance of HybridAgent on latent thoughts (Table 7), and ablate HybridLink architectures to assess the benefits of residual alignment (Table 8). **(iii) Text-Mode**: We evaluate text-based agentic tool use (Figure 11) and auditability of HybridAgent (Table 9) to verify that text-mode supports reliable environment interaction and remains informative for external monitoring/debugging. Due to page limits, we provide all analysis setups and results in Appendix G.

## 6 Related Work

**Latent Reasoning Beyond Text.** Recent works have explored applying intermediate reasoning beyond discrete language tokens into continuous hidden spaces (Hao et al., 2024; Zhu et al., 2025a; Yu et al., 2026), allowing models to perform internal computation without explicitly verbalizing every reasoning step (Geiping et al., 2026; Shen et al., 2026; Zhu et al., 2025c). Existing approaches broadly realize this idea by inputting last-layer hidden states (Saunshi et al., 2025; Tack et al., 2026), compressing textual chains of thought into latent representations (Cheng & Van Durme, 2024; Zhang et al., 2026a), or directly optimizing continuous reasoning trajectories (Tan et al., 2026; Chen et al., 2024). These methods improve reasoning efficiency and depth but make direct inspection and external interaction difficult (Korbak et al., 2025; Chen et al., 2026). HybridAgent builds on this direction and further integrates latent reasoning as a native built-in mode within a full agentic generation process, while preserving text-based auditing and interaction and keeping latent thoughts directly inspectable.

**Agentic Reasoning and Tool Use.** Frontier LLM agents (Team, 2026; GLM-5-Team et al., 2026; DeepSeek-AI, 2026) extend reasoning beyond static generation by interacting with external environments through tools (Shen, 2024; Qu et al., 2025), such as search engines (Jin et al., 2025; Zheng et al., 2025b), code interpreters (Feng et al., 2026; Bianchi et al., 2026), and APIs (Qin et al., 2024; Patil et al., 2025). A growing line of work further improves these capabilities through post-training over multi-turn tool-use trajectories (Dong et al., 2025a; Qian et al., 2025), enabling agents to plan, act, observe, and revise their reasoning (Su et al., 2025; Li et al., 2025c). Despite this progress, existing agentic frameworks remain text-centric for both model-side reasoning and environment-side interaction, incurring large generation overhead from verbalizing every intermediate step (Cuadron et al., 2025; Xu et al., 2025a). HybridAgent decouples these roles by interleaving continuous thoughts for internal reasoning and retaining text only where explicit interaction and monitoring are required.

## 7 Conclusion

We introduce HybridAgent, a unified hybrid reasoning framework that combines efficient latent reasoning with text-based inspection and tool interaction. HybridAgent enables multi-turn hybrid generation through both architectural designs and dedicated post-training algorithms on SFT and RL stages. Across 11 benchmarks and 3 model scales, HybridAgent achieves strong performance against advanced LLM agents and agentic frameworks spanning text-based and latent reasoning methods, while substantially reducing inference time, token usage, and computational cost. Overall, HybridAgent provides a scalable and efficient framework for next-generation agents that can reason beyond human language while retaining agentic interaction in environments and external inspectability.

## AI USE STATEMENT

In this work, we used LLMs as auxiliary writing tools to polish author-written text for grammar, fluency, and readability. For data curation tasks, we leverage several LLMs as expert judges to filter existing text-based SFT data and assign reasoning steps to text or latent mode, without generating or rewriting trajectory content; LLMs also appear as baseline models used in our evaluations, as described in the experiment sections. We did not use generative AI to formulate claims, propose hypotheses, design experiments, or interpret results. All scientific ideas, methodology, experiments, and analyses were developed and validated by the authors. The authors reviewed all AI-assisted content and take full responsibility for the paper. The use of LLMs does not constitute authorship.

## ETHICS STATEMENT

HybridAgent improves the efficiency of LLM agents while supporting reliable and verifiable tool interaction and external inspection. Our work does not involve human participants or the collection of new personal data, and all utilized data are derived from publicly available dataset sources used under their respective standard licenses. Experiments are conducted in controlled environments with safeguards for code execution and external tool access. We are committed to responsible research and development, and all authors have read and adhere to the ICLR Code of Ethics.

## REPRODUCIBILITY STATEMENT

We describe the detailed architecture of HybridAgent, the HybridSFT and HybridRL training procedures, and evaluation protocols in Sections 3–5. In addition, Appendix H provides dataset descriptions, baseline configurations, and training and inference settings. Formal theoretical proofs for multi-turn context continuity and hybrid policy optimization are provided in Appendices A and B. The detailed implementation is also provided in the supplementary materials.

## REFERENCES

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation, 2025. URL https://arxiv.org/abs/2503.11926.

Federico Bianchi, Yongchan Kwon, Aneesh Pappu, and James Zou. Harnessing the collective intelligence of ai agents in the wild for new discoveries, 2026. URL https://arxiv.org/abs/2606.10402.

Mahtab Bigverdi, Zelun Luo, Cheng-Yu Hsieh, Ethan Shen, Dongping Chen, Linda G. Shapiro, and Ranjay Krishna. Perception tokens enhance visual reasoning in multimodal language models, 2024. URL https://arxiv.org/abs/2412.03548.

Center for AI Safety, Scale AI, and HLE Contributors Consortium. A benchmark of expert-level academic questions to assess AI capabilities. *Nature*, 649:1139–1146, 2026. doi: 10.1038/s41586-025-09962-4. URL https://arxiv.org/abs/2501.14249.

Shuochen Chang, Tong Bai, Xiaofeng Zhang, Qianli Ma, Qingyang Liu, Zhaohe Liao, Yibo Miao, and Li Niu. Unlocking the black box of latent reasoning: An interpretability-guided approach to intervention. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34019–34032, 2026.

Haolin Chen, Yihao Feng, Zuxin Liu, Weiran Yao, Akshara Prabhakar, Shelby Heinecke, Ricky Ho, Phil Mui, Silvio Savarese, Caiming Xiong, et al. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding. *arXiv preprint arXiv:2411.04282*, 2024.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2505.16782*, 2025.

Xinghao Chen, Chak Tou Leong, Guo Wenjin, Jian Wang, Wenjie Li, and Xiaoyu Shen. What makes effective supervision in latent chain-of-thought: An information-theoretic analysis. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=x21YkF1UfR.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

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

DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4443–4458, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.408. URL https://aclanthology.org/2020.acl-main.408/.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *CoRR*, abs/2505.16410, 2025a. doi: 10.48550/ARXIV.2505.16410. URL https://doi.org/10.48550/arXiv.2505.16410.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025b.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tRk1nofSmz.

Renyu Fu and Guibo Luo. Selar: Selective latent reasoning in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7073–7087, 2026.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2026.

GLM-5-Team et al. GLM-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Nikita Gupta, Riju Chatterjee, Lukas Haas, Connie Tao, Andrew Wang, Chang Liu, Hidekazu Oiwa, Elena Gribovskaya, Jan Ackermann, John Blitzer, Sasha Goldshtein, and Dipanjan Das. Deepsearchqa: Bridging the comprehensiveness gap for deep research agents, 2026. URL https://arxiv.org/abs/2601.20975.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

HuggingFaceH4. Math-500 dataset. https://huggingface.co/datasets/HuggingFaceH4/MATH-500, 2023.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, et al. Chain of thought monitorability: A new and fragile opportunity for ai safety. *arXiv preprint arXiv:2507.11473*, 2025.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. Measuring ai ability to complete long software tasks. *Advances in Neural Information Processing Systems*, 38:92213–92266, 2026.

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. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.276. URL https://aclanthology.org/2025.emnlp-main.276/.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=7LKKHBAMzH.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*, 2025c.

Fangrui Lv, Lei Wang, Ruixin Hong, Yong Du, Xiangyu Wu, Tingting Gao, Guorui Zhou, and Changshui Zhang. Beyond tokens: Dynamic latent reasoning via semantic residual refinement. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 32383–32391. AAAI Press, 2026. doi: 10.1609/AAAI.V40I38.40513. URL https://doi.org/10.1609/aaai.v40i38.40513.

math ai. AIME 2025 dataset. https://huggingface.co/datasets/math-ai/aime25, 2025.

MathArena. Aime 2026 dataset. https://huggingface.co/datasets/MathArena/aime_2026, 2026.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Alex Ning, Yen-Ling Kuo, and Gabe Gomes. Learning when to stop: Adaptive latent reasoning via reinforcement learning. *arXiv preprint arXiv:2511.21581*, 2025.

OpenAI. GPT-5.6 Luna. OpenAI API model documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Accessed: 2026-09-16.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Thinh Pham, Nguyen Phan Nguyen, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. SealQA: Raising the bar for reasoning in search-augmented language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zWb7ueH16c.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=eOLdGbXT6t.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-rong Wen. Tool learning with large language models: a survey. *Frontiers of Computer Science*, 19(8), January 2025. ISSN 2095-2236. doi: 10.1007/s11704-024-40678-2. URL http://dx.doi.org/10.1007/s11704-024-40678-2.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, volume 2025, pp. 14855–14881, 2025.

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 Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Xuan Shen, Yizhou Wang, Yufa Zhou, Xiangxi Shi, Pu Zhao, Yanzhi Wang, and Jiuxiang Gu. Efficient reasoning with hidden thinking. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Hn81xuLaXY.

Zhuocheng Shen. Llm with tools: A survey, 2024. URL https://arxiv.org/abs/2409.18807.

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. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Dachuan Shi, Abedelkadir Asi, Keying Li, Xiangchi Yuan, Leyan Pan, Wenke Lee, and Wen Xiao. Swireasoning: Switch-thinking in latent and explicit for pareto-superior reasoning LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=t33kMzEAg8.

David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 1(11):6, 2025.

Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, Yi Dong, Evelina Bakhturina, Tao Yu, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Toolorchestra: Elevating intelligence via efficient model and tool orchestration, 2025. URL https://arxiv.org/abs/2511.21689.

Theodore Sumers, Shunyu Yao, Karthik R Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=1i6ZCvflQJ. Survey Certification, Featured Certification.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, Haoyun Zhang, Jeffrey Lin, et al. Agents' last exam. *arXiv preprint arXiv:2606.05405*, 2026.

Jihoon Tack, Jack Lanchantin, Jane Dwivedi-Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason E Weston, and Xian Li. Llm pretraining with continuous concepts. In *International Conference on Learning Representations*, volume 2026, pp. 54214–54231, 2026.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *Advances in Neural Information Processing Systems*, 38:4646–4668, 2026.

Kimi Team. Kimi K3: open frontier intelligence. *CoRR*, abs/2607.24653, 2026. doi: 10.48550/ARXIV.2607.24653. URL https://doi.org/10.48550/arXiv.2607.24653.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond images and language. *arXiv preprint arXiv:2511.21395*, 2025.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. WebWalker: Benchmarking LLMs in web traversal. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10290–10305, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.508. URL https://aclanthology.org/2025.acl-long.508/.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less, 2025a. URL https://arxiv.org/abs/2502.18600.

Xin Xu, Tong Yu, Xiang Chen, Haoliang Wang, Julian McAuley, and Saayan Mitra. Thinkrouter: Efficient reasoning via routing thinking between latent and discrete spaces. *arXiv preprint arXiv:2602.11683*, 2026.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1137. URL https://aclanthology.org/2025.acl-long.1137/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Hang Yang, Hao Chen, Hui Guo, Yineng Chen, Ching-Sheng Lin, Shu Hu, Jinrong Hu, Xi Wu, and Xin Wang. Llm-medqa: Enhancing medical question answering through case studies in large language models. *arXiv preprint arXiv:2501.05464*, 2024.

Jiayu Yang, Chao Chen, Shengen Wu, Yinhong Liu, Yuxuan Fan, Lujundong Li, Songning Lai, Chengwei Qin, and Zhijiang Guo. Demystifying hidden-state recurrence: Switchable latent reasoning with on-policy reinforcement learning. *arXiv preprint arXiv:2606.13106*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Xinwu Ye, Yicheng Mao, Jia Zhang, Yimeng Liu, Li Hao, Fang Wu, Zhiwei Li, Yuxuan Liao, Zehong Wang, Zhiyuan Liu, Zhenfei Yin, Li Yuan, Philip Torr, Huan Sun, xiangxiang Zeng, Mengdi Wang, Le Cong, Shenghua Gao, and Xiangru Tang. Latentchem: From textual cot to latent thinking in chemical reasoning. In *Workshop on Latent & Implicit Thinking – Going Beyond CoT Reasoning*, 2026. URL https://openreview.net/forum?id=ToCIS6xUwF.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Xinlei Yu, Zhangquan Chen, Yongbo He, Tianyu Fu, Guanting Dong, Cheng Yang, Chengming Xu, Yue Ma, Xiaobin Hu, Zhe Cao, et al. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv preprint arXiv:2604.02029*, 2026.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid latent reasoning via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=LjtgTpWH71.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. In *International Conference on Learning Representations*, volume 2026, pp. 22555–22588, 2026a.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, volume 2026, pp. 104223–104294, 2026b.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ByQdHPGKgU.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, 2025b.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In *ICML 2025 Workshop on Methods and Opportunities at Small Scale*, 2025a. URL https://openreview.net/forum?id=1cD9iO5Isv.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, et al. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025b.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025c.

# TABLE OF CONTENTS

**A Theoretical Analyses of Multi-turn Continuation** **17**

**B Theoretical Analyses of Hybrid Reinforcement Learning** **19**

B.1 Formulation of Hybrid Policy Factorization . . . . . 19

B.2 Hybrid Policy Optimization . . . . . 20

**C Additional Related Work** **21**

**D Hybrid Data Curation** **22**

**E Latent Interpretability Evaluation (Figure 2 and 9)** **23**

**F Additional Experiments** **24**

F.1 Experiments on 4B-Scale HybridAgent . . . . . 24

F.2 Cost Analyses on HybridAgent . . . . . 24

**G In-depth Analyses on HybridAgent** **25**

G.1 Hybrid Training: HybridSFT and HybridRL . . . . . 25

G.2 HybridAgent Latent Mode: Causal Reliance and HybridLink Design . . . . . 26

G.3 HybridAgent Text Mode: Tool Use and Auditability . . . . . 27

**H Experiment Setups** **29**

H.1 Benchmark and Evaluation Details . . . . . 29

H.2 Baseline Details . . . . . 30

H.3 Implementation Details . . . . . 31

**I Prompt Template** **33**

**J Case Study of HybridAgent** **35**

# Appendix

## A Theoretical Analyses of Multi-turn Continuation

Following our notation in Section 2, denote the input sequence $E_{\leq t} = [e_1, \ldots, e_t]$ and working memory $\mathcal{M}_t = [K_{\leq t}, V_{\leq t}]$ at step $t$, where $K$ and $V$ collect the keys and values across all layers.

**Theorem A.1** (Lossless Multi-Turn Working Memory)**.** *Given the multi-turn generation process in Section 3.2, consider the fixed HybridAgent $f_\theta$ and HybridLink $\mathcal{R}$ under consistent inference settings. At step $t$, let $h_{t+1}$ be the hidden output obtained by processing $e_{t+1}$ with the complete working memory $\mathcal{M}_t$. Then,*

$$\underbrace{h_{t+1}}_{\text{process } e_{t+1} \text{ using } \mathcal{M}_t} = \underbrace{f_\theta([E_{\leq t}; e_{t+1}])}_{\text{apply the complete hybrid prefix}} . \tag{12}$$

*Thus, working memory preserves the text, latent, and tool observations for continued reasoning, without converting latent thoughts into text.*

*Remark* A.2. Theorem A.1 shows that retaining complete working memory preserves the effect of the full hybrid history on subsequent generation across turns. This justifies resuming HybridAgent after tool interactions without reconstructing earlier turns or converting latent thoughts into text.

*Proof.* Fix $f_\theta$ and HybridLink $\mathcal{R}$, and use the same deterministic causal operations, masks, and positional encoding, preserving all existing KV entries. Each cached forward run produces both a hidden output and an updated memory:

$$\underbrace{(e_{t+1}, \mathcal{M}_t)}_{\text{next input and preceding context}} \xrightarrow{\text{cached forward run}} \underbrace{(h_{t+1}, \mathcal{M}_{t+1})}_{\text{hidden output and updated memory}} .$$

The hidden output determines the next text distribution or latent input, while the updated memory provides the context for the following forward run. Here, layer and head indices are omitted for notation simplicity. We compare a memory-based forward run with a full-prefix forward run under the same model and inference settings.

*Base case ($t = 1$).* Both forward runs start with empty memory and the same input $e_1$. At the first layer, they produce the same queries, keys, values, and attention outputs. The remaining token-wise operations produce the same input to the next layer. Repeating this argument through all layers gives

$$\mathcal{M}_1 = [K_1, V_1].$$

Both forward runs also produce the same final hidden output, $h_1 = f_\theta([e_1])$.

*Induction hypothesis.* Suppose that, after processing step $t$, the stored keys and values are equal to those computed by a full-prefix forward run:

$$\underbrace{\mathcal{M}_t}_{\text{stored KV memory}} = \underbrace{[K_{\leq t}, V_{\leq t}]}_{\text{KV computed from the complete prefix } E_{\leq t}} .$$

*Induction step ($t \to t + 1$).* Causality and unchanged positional encoding preserve the earlier keys and values when $e_{t+1}$ is appended. Both forward runs start the first layer with the same input $e_{t+1}$.

If their inputs to the current layer are equal, each head computes the same query, new keys and values. The earlier KV entries are equal by the induction hypothesis. With the same attention mask, each head therefore produces the same attention output. The remaining token-wise operations produce the same input to the following layer.

Repeating this implication through all layers gives the same final hidden output:

$$h_{t+1} = f_\theta([E_{\leq t}; e_{t+1}]).$$

It also gives the same new keys and values, $K_{t+1}, V_{t+1}$. Appending them to the stored memory and substituting the induction hypothesis yields

$$\begin{aligned}\mathcal{M}_{t+1} &= \mathcal{M}_t \oplus [K_{t+1}, V_{t+1}] \\ &= [K_{\leq t}, V_{\leq t}] \oplus [K_{t+1}, V_{t+1}] \\ &= [K_{\leq t+1}, V_{\leq t+1}].\end{aligned}$$

Thus, the updated memory satisfies the induction hypothesis at $t+1$, while the hidden-output equality establishes Eq. 12. By induction, at every input position, the cached forward run produces the same hidden output and updated KV memory as the full-prefix forward run.

At the end of each turn $j \in \{1, \ldots, n\}$, let $t = |x| + \sum_{i=1}^{j} |\tau^{(i)}|$ count all input slots processed so far, including the initial context and all text, latent, and tool-observation inputs. Starting from $\mathcal{M}^{(0)} = [K_{\leq |x|}, V_{\leq |x|}]$, repeated concatenation gives

$$\begin{aligned}\mathcal{M}^{(j)} &= \mathcal{M}^{(0)} \oplus [K_{|x|+1:t}, V_{|x|+1:t}] \\ &= [K_{\leq |x|}, V_{\leq |x|}] \oplus [K_{|x|+1:t}, V_{|x|+1:t}] \\ &= [K_{\leq t}, V_{\leq t}].\end{aligned}$$

Each new entry is appended exactly once, so turn boundaries do not change the memory used by subsequent forward runs.

Finally, the established hidden-state equality gives

$$\begin{aligned}\mathrm{Softmax}(W_{\mathrm{out}} h_t) &= \mathrm{Softmax}(W_{\mathrm{out}} f_\theta(E_{\leq t})), \\ \mathcal{R}(h_t) &= \mathcal{R}(f_\theta(E_{\leq t})).\end{aligned}$$

where we retain the corresponding hidden states and tool observations, while using the same random draws for sampling and tool execution in both runs. Then, both runs process the same next input with the same memory and reasoning mode, producing the same hidden output and updated KV memory. Repeating this argument step by step gives identical subsequent hybrid sequences under the shared random draws, and hence identical generation probabilities. Therefore, working memory preserves the effect of the complete hybrid history across turns, without reconstructing latent thoughts as text.

□

# B THEORETICAL ANALYSES OF HYBRID REINFORCEMENT LEARNING

## B.1 FORMULATION OF HYBRID POLICY FACTORIZATION

Following our notation in Algorithm 1, denote $\tau = (z, y, o)$ as a hybrid trajectory, where $\mathcal{T}$ is the set of sampled text positions. Here, $\pi_\theta(\tau \mid x)$ denotes the likelihood of the agent's outputs along trajectory $\tau$, given input $x$. Meanwhile, tool observations are generated by the environment and serve as context for subsequent model decisions. We formalize the factorization of the hybrid policy below.

**Proposition B.1** (Restatement of Hybrid Policy Factorization)**.** *Let $\tau$ be a hybrid trajectory generated by the policy $\pi_\theta$ (induced by $f_\theta$) with fixed $\mathcal{R}$. Then the hybrid policy factorizes as:*

$$\pi_\theta(\tau \mid x) \;=\; \prod_{t \in \mathcal{T}} \pi_\theta\big(y_t \mid x,\, \tau_{<t}\big), \tag{13}$$

*where $\mathcal{T}$ is the set of sampled text positions of trajectory $\tau$.*

*Proof.* Fix $\theta$ and a realized trajectory $\tau$. At a latent continuation, we have $\Pr(e_t = \mathcal{R}(h_{t-1}) \mid x, \tau_{<t}) = 1$ with a fixed HybridLink $\mathcal{R}$. Deterministic reasoning mode transitions (without sampling) also have conditional mass 1. The probability chain rule thus gives

$$\pi_\theta(\tau \mid x) = \left[\prod_{t \in \mathcal{T}} \pi_\theta(y_t \mid x, \tau_{<t})\right] \left[\prod_{t:\ \text{deterministic steps}} 1\right] = \prod_{t \in \mathcal{T}} \pi_\theta(y_t \mid x, \tau_{<t}),$$

where deterministic steps receive the probability of 1.

□

*Remark* B.2. Proposition B.1 shows that the hybrid policy factorizes over sampled text positions, with each text-token probability conditioned on the complete preceding hybrid context, including text, latent thoughts, and tool observations. Building on this factorization formulation, we show in the next subsection (Theorem B.3) how exact rollout replay enables HybridRL's policy updates to assign credit to complete recorded hybrid trajectories.

### B.2 HYBRID POLICY OPTIMIZATION

Recall that we collect $\{\tau_i = (z_i, y_i, o_i)\}_{i=1}^{G}$, with group size $G \geq 2$, using the rollout policy $\pi_{\theta_{\text{old}}}$. During replay, $\pi_\theta(y_{i,t} \mid x, \tau_{i,<t})$ scores the recorded prefix with the current model across step $t$; $\pi_\theta(\tau_i \mid x)$ is the product in Eq. 13 under this same evaluation. Let $r_i = r(\tau_i)$ be the reward, and $A_i$ be the fixed group-normalized advantage. Let $\mathcal{T}_i$ denote the sampled text positions of $\tau_i$. Define

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x, \tau_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, \tau_{i,<t})}, \tag{14}$$

$$\mathcal{L}_{\text{RL}}(\theta) = -\frac{1}{\sum_i |\mathcal{T}_i|} \sum_i \sum_{t \in \mathcal{T}_i} \min\{\rho_{i,t}(\theta)A_i, \text{clip}(\rho_{i,t}(\theta))A_i\}\,. \tag{15}$$

Here, $\text{clip}(\rho)$ restricts $\rho$ to $[1-\epsilon, 1+\epsilon]$, where $0 < \epsilon < 1$. Thus, ratios sufficiently close to 1 are unchanged: $\text{clip}(\rho) = \rho$ whenever $|\rho - 1| < \epsilon$. In the following theorem, we connect this objective to our hybrid policy optimization over complete recorded hybrid trajectories.

**Theorem B.3** (Gradient Equivalence in Hybrid Policy Optimization)**.** *Based on the rollout replay (Section 4.2), record each trajectory's latent inputs and tool observations. Under Theorem A.1 and the replay conditions in Algorithm 1, the accumulated text-token gradient is equivalent to the advantage-weighted likelihood gradient for the complete hybrid trajectories:*

$$-\nabla_\theta \mathcal{L}_{\text{RL}}(\theta_{\text{old}}) = \frac{1}{\sum_i |\mathcal{T}_i|} \sum_i A_i \nabla_\theta \log \pi_\theta(\tau_i \mid x)\Big|_{\theta=\theta_{\text{old}}}. \tag{16}$$

*Therefore, the text-token update assigns advantage-weighted policy credit to optimize the complete recorded hybrid trajectories.*

*Remark* B.4. At $\theta = \theta_{\text{old}}$, HybridRL produces the same policy update as optimizing the advantage-weighted recorded-trajectory objective. By preserving the context that produced each reward, exact rollout replay enables the backbone to learn from complete recorded hybrid trajectories.

*Proof.* For trajectory $\tau_i$, we omit the sample index on $E$ and $e$ for notation simplicity. At step $t$ of the $i$-th hybrid trajectory $\tau_i$, the subsequent input can be denoted by

$$e_t = \begin{cases} \text{Emb}(y_{i,t}), & \text{generated text token,} \\ z_{i,t}, & \text{recorded latent input,} \\ \text{Emb}(o_{i,t}), & \text{tool observation.} \end{cases}$$

During replay, $z_{i,t}$ denotes the latent input recorded under $\theta_{\text{old}}$ and is fixed when differentiating with respect to the current parameters $\theta$, leading to $\frac{\partial z_{i,t}}{\partial \theta} = 0$. Here, although the latent inputs are fixed, $h_{t-1} = f_\theta(E_{\leq t-1})$ remains parameter-dependent and retains the chain-rule contribution $\left(\frac{\partial h_{t-1}}{\partial \theta}\right)^\top \nabla_{h_{t-1}} \log \pi_\theta(y_{i,t} \mid x, \tau_{i,<t})$, allowing trajectory-level rewards to optimize the backbone's processing of the complete recorded hybrid context, while keeping the latent inputs fixed.

Then, by causality and Theorem A.1, full-sequence replay scores each sampled text token at position $t \in \mathcal{T}_i$ using the hidden state $h_{t-1} = f_\theta(E_{\leq t-1})$ computed from the recorded hybrid prefix. With this prefix, taking the logarithm and differentiating Eq. 13 gives

$$\nabla_\theta \log \pi_\theta(\tau_i \mid x) = \sum_{t \in \mathcal{T}_i} \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, \tau_{i,<t}), \tag{17}$$

which expresses the complete recorded-trajectory score as a sum of text-token scores. To connect this equality to the implemented loss, we next show that each token-ratio gradient equals its text-token score at the rollout policy $\theta_{\text{old}}$.

The rollout probability in the denominator of Eq. 14 is fixed during differentiation. Therefore,

$$\begin{aligned}\nabla_\theta \rho_{i,t}(\theta) &= \frac{\nabla_\theta \pi_\theta(y_{i,t} \mid x, \tau_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, \tau_{i,<t})} \\ &= \frac{\pi_\theta(y_{i,t} \mid x, \tau_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, \tau_{i,<t})} \frac{\nabla_\theta \pi_\theta(y_{i,t} \mid x, \tau_{i,<t})}{\pi_\theta(y_{i,t} \mid x, \tau_{i,<t})} \\ &= \rho_{i,t}(\theta)\, \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, \tau_{i,<t}),\end{aligned}$$

where the last equality uses $\nabla_\theta \log p = (\nabla_\theta p)/p$. At $\theta = \theta_{\text{old}}$, exact replay reproduces the rollout probability, giving $\rho_{i,t}(\theta_{\text{old}}) = 1$. Therefore, evaluating the derivative yields

$$\nabla_\theta \rho_{i,t}(\theta)|_{\theta=\theta_{\text{old}}} = \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, \tau_{i,<t})|_{\theta=\theta_{\text{old}}}.$$

Since clipping leaves ratios near 1 unchanged, each summand in Eq. 15 equals $\rho_{i,t}(\theta)A_i$ locally, leading to

$$\begin{aligned}-\nabla_\theta \mathcal{L}_{\text{RL}}(\theta)|_{\theta=\theta_{\text{old}}} &= \frac{1}{\sum_i |\mathcal{T}_i|} \sum_i A_i \sum_{t \in \mathcal{T}_i} \nabla_\theta \rho_{i,t}(\theta) \Big|_{\theta=\theta_{\text{old}}} \\ &= \frac{1}{\sum_i |\mathcal{T}_i|} \sum_i A_i \sum_{t \in \mathcal{T}_i} \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, \tau_{i,<t}) \Big|_{\theta=\theta_{\text{old}}} \\ &= \frac{1}{\sum_i |\mathcal{T}_i|} \sum_i A_i \nabla_\theta \log \pi_\theta(\tau_i \mid x) \Big|_{\theta=\theta_{\text{old}}},\end{aligned}$$

where the second equality uses the token-ratio derivative at $\theta_{\text{old}}$, and the final equality uses Eq. 17.

□

## C Additional Related Work

**Adaptive Reasoning on Text and Latent Space.** Beyond fixed-length chains of thought, recent work lets LLMs adapt how long and in which medium they reason (Bigverdi et al., 2024; Lv et al., 2026; Yang et al., 2026; Ye et al., 2026), for example by learning when to stop latent reasoning or deciding whether explicit deliberation is needed (Ning et al., 2025). Several methods interleave latent reasoning with text decoding (Yue et al., 2025), using confidence or entropy signals (Xu et al., 2026) to shift into continuous representations at uncertain steps and explore alternative reasoning paths (Fu & Luo, 2026; Shi et al., 2026). However, existing methods largely operate within one reasoning pass or switch modes only at single-turn granularity, and their latent states remain difficult to inspect directly. Orthogonal to this line, HybridAgent uses latent mode for efficient deeper reasoning, carries latent context across multiple turns through continuous working memory, keeps latent thoughts directly readable, and preserves the agentic tool-use abilities for long-horizon tasks.

## D HYBRID DATA CURATION

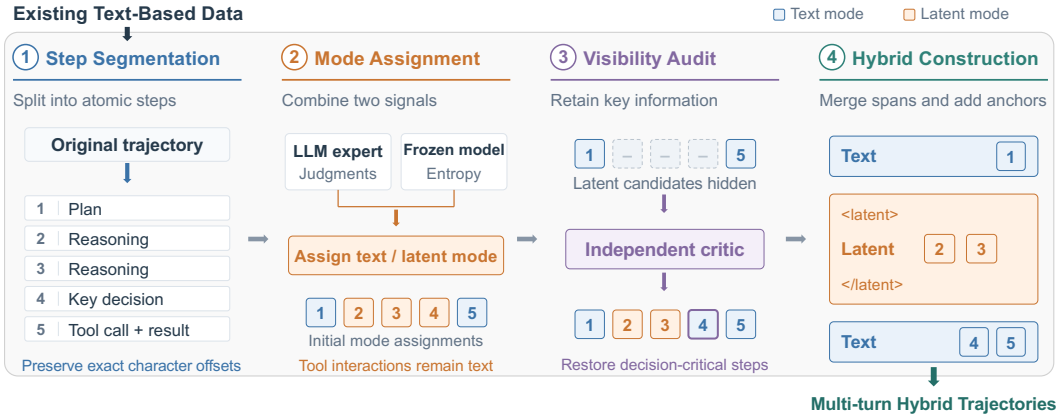


Figure 8: **Hybrid Data Curation Pipeline.** We convert text-based demonstrations into multi-turn hybrid trajectories using complementary signals from LLM judgments and model entropy to assign reasoning modes, followed by a visibility audit that preserves decision-critical steps in text mode.

We curate the hybrid training data by converting existing multi-turn agentic demonstrations (Yu et al., 2025; Dong et al., 2025b) into 54k hybrid trajectories for text and latent supervision. Specifically, an automatic annotation pipeline is designed to assign each source CoT step to either text or latent mode, based on complementary signals from external LLM judges and the target model's entropy-based confidence, with no human labeling needed. Accordingly, high-confidence reasoning steps that the model can handle reliably are assigned as latent targets, while audit-critical reasoning and tool-interaction steps are retained as text targets. We describe the detailed curation pipeline below.

**1 Step Segmentation.** We first split each source trajectory into atomic reasoning steps using a deterministic procedure, recording their exact character offsets. These offsets allow subsequent annotations to identify and group steps while preserving the original content and ordering.

**2 Mode Assignment.** We then leverage an external LLM expert (GPT-5.6-Luna) to assign each reasoning step to text or latent mode, with tool interactions and key steps needed for external inspection retained in text mode (See Prompts in Appendix I). Meanwhile, we complement the external selection with the target model's prediction entropy. We teacher-force the original textual trajectory through a frozen base model and compute the mean token-level entropy for each step $s$:

$$H_s = -\frac{1}{|\mathcal{I}_s|} \sum_{t \in \mathcal{I}_s} \sum_{v \in \mathcal{V}} f_\theta(v \mid x, \tau_{<t}) \log f_\theta(v \mid x, \tau_{<t}), \tag{18}$$

where $f_\theta(v \mid x, \tau_{<t})$ denotes the frozen base model's next-token probability after the vocabulary projection and softmax. Here, $x$ is the input question, $\tau_{<t}$ contains preceding reasoning and tool observations, $\mathcal{I}_s$ indexes the tokens in step $s$, and $\mathcal{V}$ is the vocabulary. Low-entropy steps become latent candidates, while high-entropy steps remain in text mode. Initial mode assignments combining the two complementary signals are then refined through the visibility audit below.

**3 Visibility Audit.** An independent LLM critic next examines the visible version of each annotated trajectory, with candidate latent spans hidden. The critic assesses whether the remaining text and tool interactions provide sufficient information to follow the agent's decisions. When decision-critical information is missing, the corresponding source steps are restored to text mode. This audit helps preserve the key information needed for external debugging and inspection.

**4 Hybrid Trajectory Construction.** We merge adjacent latent-assigned steps into continuous spans and programmatically insert the special anchors `<latent>` and `</latent>`. Tool calls and observations remain in text form. Finally, we verify that removing the inserted anchors recovers the original trajectory byte for byte. This consistency check ensures that the pipeline annotates reasoning modes without rewriting or altering the source demonstrations.

# E LATENT INTERPRETABILITY EVALUATION (FIGURE 2 AND 9)

In this section, we describe the evaluation setup and metrics used to measure the readability and context specificity of latent readouts in Figures 2 and 9.

**Probe Setup.** We evaluate HybridAgent across 200 questions spanning downstream tasks, including AIME25/26 (math ai, 2025; MathArena, 2026), MATH-500 (HuggingFaceH4, 2023), MuSiQue (Trivedi et al., 2022), GAIA (Mialon et al., 2024), and DSQA (Gupta et al., 2026). Figure 2 compares the final HybridRL checkpoints of HybridAgent at 4B, 8B, and 14B scales. Figure 9 compares format activation, HybridLink activation, self-conditioned HybridSFT, and Coconut on the shared Qwen3-14B-Base backbone. Generation uses temperature 0.6, top-p 0.95, top-k 20, and a fixed random seed. Each unit inside the plots is a *latent span*: a maximal sequence of consecutive latent steps. Each plotted point represents one span, and rings mark the means of per-span scores for each model or variant. The annotation $n = 200$ refers to total number of questions.

**Readability.** For a latent span $b$ containing $L_b$ latent inputs $z_{b,t}$, we measure *readability score* through their average nearest-token alignment, following Eq. 5:

$$\mathrm{Read}(b) = \frac{1}{L_b} \sum_{t=1}^{L_b} \max_{v \in \mathcal{V}} \cos(z_{b,t}, \mathrm{Emb}(v)) \,, \tag{19}$$

where $\mathcal{V}$ is the vocabulary and $\mathrm{Emb}(v)$ is the input embedding of token $v$. A score of 1 indicates that every latent input shares the direction of a vocabulary embedding, corresponding to the "exact token" reference. This alignment serves as a geometric proxy for readability. To calibrate the noise reference, we apply the same readout to 2,000 Gaussian random vectors and use their mean nearest-token similarity as the threshold $\tau_{\mathrm{noise}}$. Spans above this threshold exhibit stronger average token alignment than the Gaussian reference.

**Context Specificity.** Context specificity measures whether a span's pseudo-text is more relevant to its originating question than to other questions from the same task. Let $P_b$ denote the pseudo-text obtained by nearest-token readout of span $b$, as in Eq. 5. Let $W(s)$ extract the content words from string $s$: lower-cased alphanumeric tokens longer than two characters, with stop words removed. We use Jaccard overlap, $J(A, B) = |A \cap B|/|A \cup B|$. For a task-specific question set $\{q_1, \ldots, q_M\}$ with $M \geq 2$, let $i(b)$ identify the question that produced span $b$. We define the *context specificity score* as:

$$\mathrm{Spec}(b) = \frac{J\big(W(P_b), W(q_{i(b)})\big)}{\dfrac{1}{M-1} \displaystyle\sum_{\substack{j=1 \\ j \neq i(b)}}^{M} J(W(P_b), W(q_j))}, \tag{20}$$

when the denominator is positive. A score of $1\times$ indicates equal overlap, while higher scores above $1\times$ indicate greater lexical relevance to the originating question. We exclude readouts without content words and plot positive scores on a logarithmic axis.

**Meaningful Region.** The shaded region contains spans whose readability exceeds $\tau_{\mathrm{noise}}$ and whose context specificity exceeds 1. The two axes are complementary: readability measures alignment with token embeddings, while context specificity measures relevance to the originating question. Together, they provide evidence of readable, question-specific latent readouts that support external inspection.

# F Additional Experiments

## F.1 Experiments on 4B-Scale HybridAgent

Table 3: **4B-Scale HybridAgent on Math and Knowledge-Intensive Search.** We evaluate performance across both SFT and RL stages. AIME25/26 are reported on avg@16 accuracy.

| Method | AIME25 | AIME26 | MATH500 | GPQA-D |
|---|---|---|---|---|
| **Qwen3-4B-Base** | | | | |
| Coconut | 10.6 | 13.3 | 40.6 | 17.2 |
| SoftCoT | 14.8 | 18.5 | 55.2 | 26.3 |
| Standard Text SFT | 19.0 | 22.7 | 77.4 | 31.8 |
| **HybridAgent (SFT)** | 22.5 | 23.3 | 76.8 | 30.8 |
| GRPO | 28.1 | 35.6 | 80.6 | 36.9 |
| ARPO | 31.3 | 36.3 | 81.8 | 38.4 |
| **HybridAgent (RL)** | **35.2** | **37.5** | **82.2** | **40.9** |

To examine the effectiveness of hybrid reasoning at smaller model scales, we evaluate HybridAgent and corresponding text-based and latent baselines on Qwen3-4B-Base. As shown in Table 3, HybridSFT outperforms both latent baselines across all four benchmarks and improves over standard text SFT on AIME25 and AIME26. HybridRL further improves performance across all four benchmarks, achieving the best results and exceeding ARPO by an average of 2.0 percentage points. Together with 8B and 14B comparisons in Table 1 and 2, these results show that the benefits of hybrid reasoning can extend to different model scales of HybridAgent across downstream reasoning tasks.

## F.2 Cost Analyses on HybridAgent

Table 4: **Cost Analyses on HybridAgent.** We compare HybridAgent with text agents under the same backbones, reporting compute per question and estimated monetary cost per 1M questions. Each text or latent generation step is charged as one full backbone forward pass. Monetary costs assume \$2.50 per H100-hour and 10% utilization of peak BF16 throughput (approximately \$0.007 per PFLOP).

| Model Backbone | Text Agent (TFLOP) | HybridAgent (TFLOP) | Cost / 1M questions (Text Agent → HybridAgent) | Cheaper by |
|---|---|---|---|---|
| Qwen3-4B-Base | 51.2 | 13.9 | \$359 → \$98 | 3.7× |
| Qwen3-8B-Base | 62.5 | 17.2 | \$439 → \$121 | 3.6× |
| Qwen3-14B-Base | 66.9 | 25.1 | \$470 → \$176 | 2.7× |

**Cost Analyses on HybridAgent.** To examine whether the efficiency of hybrid generation translates into lower computational cost, we compare HybridAgent with matched text agents across 4B, 8B, and 14B backbones. We estimate decoding compute by charging every text or latent step a full backbone forward pass, together with the cost of vocabulary projection or the HybridLink adapter, respectively. Step counts are averaged over matched evaluations with the same questions, generation budgets, and tool-call limits. We further estimate monetary costs assuming \$2.50 per H100-hour and 10% utilization of peak BF16 throughput. These estimates cover generation compute and exclude prompt prefill, context-dependent attention, and external tool execution. As shown in Table 4, HybridAgent consistently reduces estimated decoding compute across all three model scales, from 51.2–66.9 TFLOP to 13.9–25.1 TFLOP per question, corresponding to reductions of 62.5%–72.9%. Under the stated pricing and utilization assumptions, the estimated cost of answering one million questions decreases from \$359 to \$98 at 4B, from \$439 to \$121 at 8B, and from \$470 to \$176 at 14B, yielding cost reduction factors of 3.7×, 3.6×, and 2.7×, respectively. Together with the shorter generation trajectories in Figure 7, these results demonstrate that hybrid reasoning reduces overall decoding compute while explicitly accounting for the backbone computation incurred by latent steps.

# G In-depth Analyses on HybridAgent

## G.1 Hybrid Training: HybridSFT and HybridRL

Table 5: **Effectiveness of Self-Conditioned HybridSFT.** We compare successive HybridSFT stages with latent and text training baselines using the same Qwen3-14B-Base backbone.

| Method | AIME25 | AIME26 | MATH500 |
|---|---|---|---|
| Coconut | 16.3 | 15.2 | 61.2 |
| Standard Text SFT | 30.8 | **31.7** | 83.8 |
| Qwen3-14B-Base | 4.9 | 3.0 | 47.0 |
| + Format Activation | 20.6 | 19.8 | 62.4 |
| + HybridLink Activation | 23.8 | 22.5 | 73.8 |
| **+ Self-Condition (HybridSFT)** | **32.5** | **31.7** | **84.2** |

**Self-Conditioning Enables Effective and Interpretable HybridSFT.** To understand why self-conditioned training is necessary for HybridAgent during SFT, we compare HybridSFT against variants with only format and HybridLink activations as well as latent SFT baselines like Coconut, from both performance and interpretability perspectives. As shown in Table 5, although all SFT variants improve the original shared Qwen3-14B-Base backbone, activation alone remains insufficient. Self-conditioned supervision further improves performance by an average of 30.5% over HybridLink activation, bringing HybridSFT to parity with standard text SFT across all three math benchmarks.

We further probe the semantics of the latent thoughts produced by each method over 200 questions spanning all downstream domains. Figure 9 reveals a clear progression, where HybridLink activation first makes latent contents more interpretable, and the additional self-conditioned training sharpens the latent readability and semantic informativeness. We then take a closer look to examine individual test instances through detailed case studies in Appendix J. Specifically, methods with low faithfulness and specificity generally produce largely unreadable artifacts of the embedding geometry, whereas the complete HybridSFT yields clear and semantically coherent latent readouts. Overall, these analyses reveal that self-conditioned training is essential to HybridSFT for making the latent reasoning traces in HybridAgent fully transparent and directly interpretable.

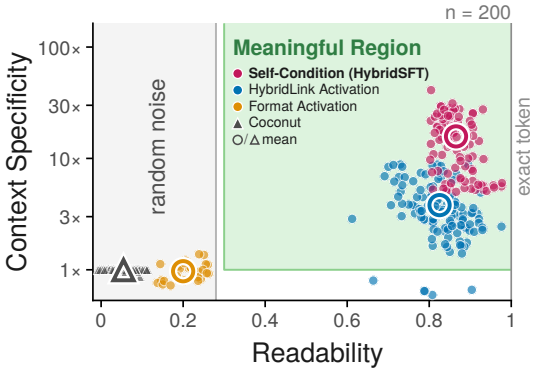


Figure 9: **Interpretability of HybridSFT.** We evaluate the readability and context-specificity across 200 downstream questions, showing how self-conditioning improves latent transparency.

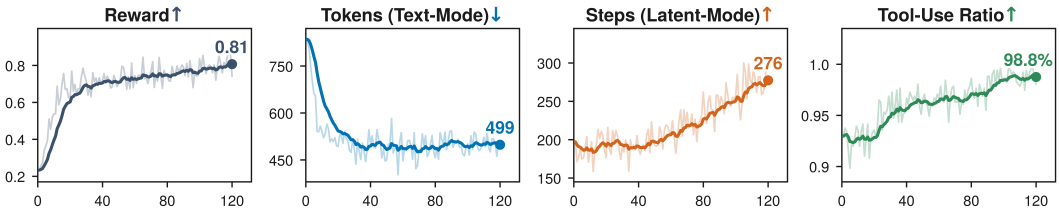


Figure 10: **Training Dynamics of HybridRL.** As training progresses, HybridAgent shifts more generation from text to latent mode while increasing reward and tool use. These trends reflect how HybridAgent adaptively learns to allocate its reasoning and tool calling efforts across the two modes.

**Training Dynamics of HybridRL.** We analyze HybridAgent's training behaviors to examine how it dynamically shifts and allocates reasoning effort across text/latent modes under HybridRL. In

Figure 10, as the training step increases, our 14B agent progressively allocates more generation to latent mode while simultaneously strengthening its tool-invocation capability. Through training, HybridAgent adaptively learns to better utilize its two modes, shifting more reasoning into faster latent thinking while selectively exposing only key steps for external auditing and tool use through the text interface.

**Reward Shaping in HybridRL.** We ablate each reward component in Eq. 11 using HybridAgent (8B), removing one term at a time under otherwise identical setups. In Table 6, all three components provide complementary supervision. Excluding $r_{\text{correct}}$ causes the largest overall degradation, reducing performance by 6.4%-17.8%, while removing $r_{\text{format}}$ consistently lowers performance by 4.7%-6.9%. The benefit of $r_{\text{tool}}$ is most pronounced on agentic benchmarks such as GAIA and WebWalker. Since the reward is granted only to trajectories that invoke both Python and search tools, removing $r_{\text{tool}}$ has a minor effect on MATH500, where Python alone is typically sufficient, but reduces GAIA and WebWalker by 11.2% and 3.5%. Overall, each reward component provides complementary supervision for answer correctness, format structure, and effective multi-tool use.

Table 6: Ablation on HybridRL Rewards.

| **Reward Design** | MATH500 | GAIA | Webwalker |
|---|---|---|---|
| HybridRL (8B) | **84.6** | **43.7** | **42.5** |
| w/o $r_{\text{correct}}$ | 79.2 | 35.9 | 37.5 |
| w/o $r_{\text{format}}$ | 80.6 | 40.7 | 40.5 |
| w/o $r_{\text{tool}}$ | 84.2 | 38.8 | 41.0 |

### G.2 HybridAgent Latent Mode: Causal Reliance and HybridLink Design

Table 7: **Causal Interventions on HybridAgent Latent-Mode.** We report MATH500 accuracy (%) for 8B and 14B scales under different interventions on latent-mode thoughts generated by HybridAgent. Performance drops substantially when latent semantics or structure are disrupted, indicating that HybridAgent causally relies on semantically meaningful latent-mode thoughts. Here, $v$ denotes the original latent vector and $v'$ denotes the intervened counterpart. The relative perturbation strength $d$ is computed as $\|v' - v\|/\|v\|$, averaged over latent steps, where $d = 0$ means no intervention.

| **Latent-Mode Intervention** | **Setup Description** | **Strength** $d$ | **14B-Scale** | **8B-Scale** |
|---|---|---|---|---|
| Gaussian Noise ($\sigma = 0.5$) | Add perturbation with random noise of magnitude $0.5\|v\|$ before latent feedback. | 0.5 | 75.0 | 72.6 |
| Gaussian Noise ($\sigma = 2.0$) | Add perturbation with random noise of magnitude $2.0\|v\|$ before latent feedback. | 2.0 | 70.2 | 66.4 |
| Random-token Replacement | Replace $v$ with a uniformly sampled token embedding, rescaled to $\|v\|$. | 3.4 | 44.6 | 41.2 |
| Within-block Permutation | Randomly permute latent vectors within each block, preserving content but disrupting order. | 0.8 | 76.4 | 73.2 |
| Cross-problem Replacement | Replace each latent span with the corresponding latents from a different partner problem. | 1.4 | 48.6 | 44.8 |
| Mean-latent Replacement | Replace every $v$ with the mean latent vector from the original HybridAgent trajectories. | 1.0 | 58.4 | 53.6 |
| Nearest-token Replacement | Replace $v$ with its nearest token embedding, rescaled to $\|v\|$. | 0.2 | 87.2 | 82.6 |
| One-step Only for Latent Span | Force each latent span to terminate after one generation step. | – | 75.0 | 72.2 |
| Block Latent-Mode Entry | Prevent entry into latent mode, forcing the agent to reason entirely in text. | – | 78.8 | 80.4 |
| **HybridAgent (ours)** | No intervention; the agent feeds back its own generated latent vector $v$ at every latent step. | **0.0** | **88.2** | **84.6** |

**Does HybridAgent Causally Rely on its Latent-Mode Thoughts?** To examine whether latent thoughts are functionally used during hybrid reasoning, we intervene on the latent vectors generated

by 8B and 14B HybridAgent on the MATH500 downstream task. We perturb their content, ordering, and reasoning duration through controlled interventions including (i) noise injection, (ii) latent span permutation, (iii) cross-problem replacement, and (iv) early termination.

As shown in Table 7, disrupting latent thoughts consistently degrades performance across both scales. Specifically, cross-problem and random-token replacement reduce accuracy by 44.9-47.0% and 49.4-51.3%, while disrupting their ordering or restricting each latent span to one step leads to 13.4-15.0% degradation. Notably, cross-problem replacement causes substantially greater degradation than even stronger Gaussian perturbations, suggesting that performance depends on the semantic content of latent thoughts rather than perturbation magnitude alone. These analyses results on causal interventions indicate that HybridAgent functionally relies on the semantics, ordering, and sufficient reasoning depth of its latent thoughts during hybrid reasoning.

**HybridLink Design.** We next validate the effectiveness of HybridLink and its residual connection by comparing our adopted 2-layer residual design against 4 alternative architectures: (i) direct hidden-state reuse (i.e., $e_{t+1}{=}h_t$) without learned alignment, (ii) 1-layer projection, (iii) 1-layer projection with residual connection, and (iv) 2-layer projection without residual connection. We end-to-end train each HybridLink variant with the 14B backbone under the same setup and evaluate across three tasks.

As shown in Table 8, the full two-layer residual design of HybridLink yields consistent gains across all three downstream benchmarks, with the residual branch benefiting both one- and two-layer projection networks. Specifically, incorporating the residual connection contributes an average improvement of 4.3% across different layer depths. This observed trend aligns with our design motivation in Section 3.1, where the residual connection preserves the semantics of input last-layer hidden states while the learned projection aligns these representations with the input embedding space. Together, these two components make HybridLink a more effective and stable interface for hybrid training and generation.

Table 8: **Effectiveness of HybridLink.** We compare accuracy across alternative architectural designs.

| **HybridLink Design** | MATH500 | MedQA | MuSiQue |
|---|---|---|---|
| Identity (No Alignment) | 71.4 | 73.7 | 24.5 |
| 1-Layer | 83.6 | 79.7 | 28.5 |
| 1-Layer+Res. | 86.4 | 80.3 | 31.0 |
| 2-Layer | 85.8 | 81.0 | 31.5 |
| **2-Layer+Res. (ours)** | **88.2** | **81.7** | **34.5** |

### G.3 HybridAgent Text Mode: Tool Use and Auditability

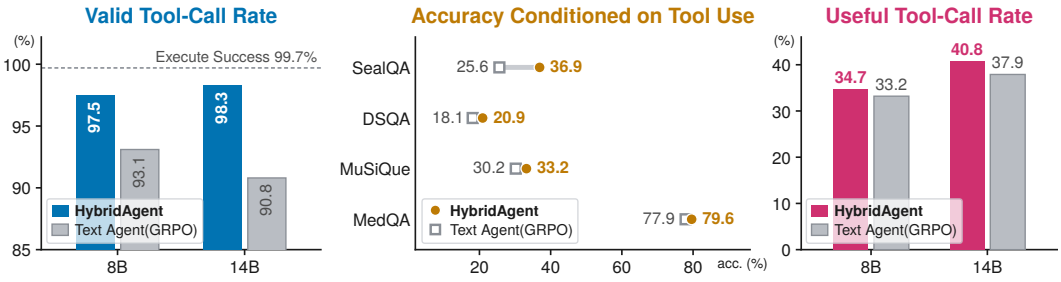


Figure 11: **Agentic Tool-Use Analysis on HybridAgent.** *Left*: Valid tool-call rate across 8B and 14B scales; "Execute Success" measures error-free execution among valid tool calls. *Middle*: Final-answer accuracy on questions where the agent performs at least one tool call. *Right*: Useful tool-call rate, measuring the fraction of tool calls whose execution results contain the final answer.

**Does HybridAgent Text-Mode Preserve Agentic Tool Use?** In Figure 11, we analyze agentic tool-calling capabilities by comparing HybridAgent with normal LLM agents trained on GRPO, examining whether hybrid reasoning preserves reliable external interactions through its text mode. Across 8B and 14B scales, HybridAgent achieves higher valid tool-call rates while matching the 99.7% execution success of valid calls, showing that hybrid reasoning preserves a reliable text-mode tool interface. More importantly, the gains extend beyond call validity, as HybridAgent improves final-answer accuracy by 2.2%-44.1% on questions where tools are invoked. The 4.5%-7.7% higher useful tool-call rate further indicates that HybridAgent not only invokes tools correctly, but also retrieves evidence that more directly contributes to solving the task. These findings suggest that

Table 9: **Monitorability and Debuggability through HybridAgent Text-Mode.** We compare HybridAgent with matched same-size text agents to examine whether its exposed text remains sufficient for external monitoring, debugging, correction, and rationale-based inspection under hybrid reasoning. All comparisons are evaluated using paired bootstrap 95% confidence intervals.

| Metric | Measurement | 8B Scale | | 14B Scale | |
|---|---|---|---|---|---|
| | | HybridAgent | Text Agent | HybridAgent | Text Agent |
| Visible Trace | Text-Mode tokens per instance | 302 | 596 | 284 | 497 |
| | Latent-Mode steps per instance | 170 | – | 183 | – |
| | Key reasoning steps per 100 visible tokens ↑ | 0.72 | 0.57 | 0.86 | 0.67 |
| Monitoring | Weak-monitor AUROC (MATH500) ↑ | 0.840 | 0.772 | 0.860 | 0.818 |
| | Strong-monitor AUROC (all tasks) ↑ | 0.961 | 0.935 | 0.989 | 0.990 |
| Debugging | First error correctly localized (%) ↑ | 99.2 | 94.2 | 98.1 | 100.0 |
| | Error diagnosis matches reference (%) ↑ | 97.6 | 93.5 | 97.2 | 96.3 |
| | Useful corrective hint / Answer leakage (%) ↑ / ↓ | 97.6 / 0.8 | 98.7 / 0.0 | 100.0 / 0.9 | 98.2 / 0.9 |
| Correction | Fix-rate gain from targeted hint vs. generic retry (Δ) ↑ | **+9.1** | +3.4 | **+5.4** | +3.5 |
| Sufficiency | Answer-agreement gain from visible text only (Δ) ↑ | **+0.25** | +0.17 | **+0.30** | +0.28 |

HybridAgent preserves the reliable and effective agentic tool use of standard LLM agents through its text mode, while exposing fewer text tokens with interleaved latent reasoning.

**Does HybridAgent Text-Mode Preserve External Auditability?** A natural question of hybrid reasoning is whether shifting computation into latent mode might potentially compromise the external auditability provided by text. To examine this, we evaluate HybridAgent against matched text agents on MATH500 and AIME25/26, covering 560 problems under identical setups. For each question, an external observer sees only the exposed text steps, tool calls, and tool outputs, while latent spans remain hidden behind their boundary markers. We evaluate four complementary dimensions following commonly adopted evaluation protocols.

- *Monitoring*: Measure whether a blind external monitor can predict rollout correctness from the input problem and visible trace generated by each agent, following Baker et al. (2025). We report AUROC using GPT-4o-mini (Hurst et al., 2024) as a weak monitor, which relies primarily on the visible trace rather than solving the problem independently, and GPT-5.6-Luna (OpenAI, 2026) as a strong monitor with more powerful task-solving capability.
- *Debugging*: Following first-error localization (Zheng et al., 2025a), a judge model (GPT-5.6-Luna) is deployed to identify the first erroneous visible step from a failed rollout and produces a corrective hint, with the diagnosis verified against the reference ground-truth solution.
- *Correction*: Each failed problem is re-attempted using the corrective hint generated during debugging, and compared against a generic correctness-only feedback baseline (Huang et al., 2024) that simply tells the agent its previous answer was incorrect and asks it to retry.
- *Rationale Sufficiency*: Following sufficiency evaluation in DeYoung et al. (2020), a weak reader (GPT-4o-mini) predicts the agent's own answer from the exposed text steps alone after removing the final answer, measuring whether the visible trace retains sufficient decision-relevant information.

As shown in Table 9, HybridAgent exposes 43%–49% fewer text-mode tokens while performing 170-183 reasoning steps in latent mode, yet the visible trace contains 26%-28% more key reasoning steps per 100 tokens. This reduction preserves external monitorability, with AUROC remaining within 0.022 of matched text agents across both weak and strong monitors. The exposed text also remains debuggable, localizing the first error in 98%-99% of failed rollouts and yielding reference-consistent diagnoses in 97%-98% of cases with below 1% answer leakage. Moreover, text-derived corrective hints produce significant fix-rate gains of 5.4 and 9.1 points at 14B and 8B, while rationale sufficiency remains comparable to or higher than text agents. As shown by the analyses above, HybridAgent uses its text mode to maintain a compact yet informative textual interface for external monitoring, debugging, and correction. This provides an important safeguard for agent generation, facilitating human auditing and intervention while preserving efficient internal reasoning.

# H EXPERIMENT SETUPS

## H.1 BENCHMARK AND EVALUATION DETAILS

We evaluate HybridAgent on 11 benchmarks spanning mathematical, knowledge-intensive, and agentic reasoning. Together, these benchmarks assess complementary capabilities in multi-step problem solving, specialized knowledge application, evidence integration, and tool-assisted information seeking. We describe each benchmark details below.

**(i) Mathematical Reasoning Tasks**

*AIME 25* (math ai, 2025): AIME 25 comprises 30 problems from the 2025 American Invitational Mathematics Examination, covering algebra, geometry, number theory, and combinatorics. Its integer-answer problems evaluate competition-level mathematical reasoning, requiring models to combine mathematical insights with precise multi-step calculations.

*AIME 26* (MathArena, 2026): AIME 26 contains 30 problems from the 2026 American Invitational Mathematics Examination, following the same competition format as AIME 25. It provides a complementary assessment of mathematical problem solving across a separate annual problem set, testing sustained reasoning and accurate numerical derivation.

*MATH500* (HuggingFaceH4, 2023): MATH500 is a 500-problem subset of the MATH test set, spanning 7 mathematical subjects and 5 difficulty levels. It evaluates math reasoning across diverse problem types, including symbolic manipulation, geometric deduction, and numerical computation.

**(ii) Knowledge-Intensive Reasoning Tasks**

*GPQA-Diamond* (Rein et al., 2023): GPQA-Diamond is a curated subset of 198 expert-written multiple-choice questions in biology, physics, and chemistry. Selected through expert and non-expert validation, these questions assess advanced scientific knowledge and reasoning that remain challenging for non-experts even with web access.

*MedQA* (Yang et al., 2024): MedQA is a multiple-choice question-answering benchmark derived from professional medical licensing examinations. The dataset contains 300 questions in total. Its questions assess medical knowledge and clinical reasoning, requiring models to interpret patient information and identify appropriate diagnoses, investigations, or treatments.

*MuSiQue* (Trivedi et al., 2022): MuSiQue is a multi-hop question-answering benchmark constructed by composing single-hop questions into connected reasoning chains of two to four hops. It evaluates the ability to combine evidence across multiple passages, with question construction designed to reduce shortcuts that bypass intermediate reasoning steps. Following Dong et al. (2025b), we use 200 test-set questions for downstream evaluation across all methods.

**(iii) Agentic Reasoning Tasks**

*General AI Assistants (GAIA)* (Mialon et al., 2024): GAIA evaluates general-purpose assistants on real-world questions involving reasoning, web browsing, information processing, and external tool use. Its three difficulty levels reflect increasingly demanding combinations of reasoning steps and tool interactions, assessing the ability to coordinate these capabilities toward a verifiable answer.

*Humanity's Last Exam (HLE)* (Center for AI Safety et al., 2026): HLE comprises expert-authored questions across a broad range of academic disciplines, including mathematics, natural sciences, and the humanities. Its multiple-choice and short-answer problems assess advanced subject knowledge and reasoning on questions designed to resist quick answers through direct web retrieval.

*WebWalker* (Wu et al., 2025): WebWalkerQA, referred to as WebWalker in our evaluations, is an agentic search benchmark requiring exploration of website subpages and integration of information from one or multiple sources. Its easy, medium, and hard categories reflect increasing traversal depth, testing the ability to locate and synthesize evidence beyond surface-level search results.

*SealQA* (Pham et al., 2026): SealQA evaluates search-augmented question answering on factual questions for which retrieved information may be noisy, conflicting, or unhelpful. We evaluate on Seal-0 and Seal-Hard, which test factual accuracy and reasoning under challenging retrieval conditions. Seal-0 targets questions selected for near-zero accuracy among the non-reasoning models used during benchmark construction; Seal-Hard includes a broader collection of difficult questions.

*DeepSearchQA (DSQA)* (Gupta et al., 2026): DeepSearchQA evaluates multi-step web research through tasks that require assembling complete answer sets from information distributed across sources. It tests search planning, evidence aggregation, entity disambiguation, and the ability to determine when sufficient information has been collected. We report both full-correct rate and F1 score to assess complete task success and the balance between answer precision and coverage.

### H.2 Baseline Details

To evaluate the effectiveness and efficiency of HybridAgent, we compare with 4 categories of methods, including (i) Text-based Agentic Methods, (ii) Latent Reasoning Methods, (iii) Frontier LLM Agents, and (iv) Agentic Frameworks. We describe the implementation details of each baseline below.

**(i) Text-based Agentic Methods**

*Standard Text SFT*: We apply standard SFT using the same training data and backbone models as HybridAgent. During evaluation, the models can invoke tools and incorporate the returned observations into subsequent reasoning. We use this baseline mostly to compare with the HybridSFT.

*Agentic RL (GRPO/ARPO)*: We apply commonly adopted agentic RL methods after standard text SFT to further train the text agent baselines. GRPO (Shao et al., 2024) optimizes the policy using group-normalized rewards from sampled responses without requiring a separate critic. ARPO (Dong et al., 2025b): ARPO combines entropy-guided adaptive rollouts with step-level advantage attribution to improve multi-turn tool use. We train these methods using the same RL training data and backbone models as HybridAgent, primarily to evaluate the effectiveness of HybridRL.

**(ii) Latent Reasoning Methods**

*Coconut* (Hao et al., 2024): Coconut performs continuous latent reasoning by directly feeding the model's last hidden state back as the next input embedding. A staged training curriculum progressively replaces textual reasoning steps with latent thoughts, teaching the model to reason in continuous space before generating a textual answer.

*SoftCoT* (Xu et al., 2025b): SoftCoT uses a lightweight frozen assistant model to generate question-dependent soft thought representations, which a trainable projection maps into the target LLM's representation space. These representations guide subsequent generation, enabling latent reasoning while keeping the assistant and target LLM frozen.

*SoftThinking* (Zhang et al., 2025): SoftThinking is a context-based method that represents intermediate reasoning as probability-weighted mixtures of vocabulary embeddings. Feeding these hidden continuous representations back into the model preserves alternatives from the next-token distribution.

We implement all latent reasoning baselines using the same backbone models and, where applicable, training settings and budgets as HybridAgent to ensure a fair comparison.

**(iii) Frontier LLM Agents**

To evaluate HybridAgent in a broader context, we compare it with current leading closed-source and open-source LLM agents, including GPT-5.6-Luna (OpenAI, 2026), Kimi-K3 (Team, 2026), GLM-5.3-Flash (GLM-5-Team et al., 2026), DeepSeek-V4.1-Flash (DeepSeek-AI, 2026), and QwQ-32B (Team, 2025), to assess both performance and inference efficiency of HybridAgent.

**(iv) Agentic Frameworks**

We further compare HybridAgent with established agentic frameworks that integrate multi-step reasoning with external tool calling and environmental interactions. We apply Qwen3-14B as the backbone for all these frameworks, with the same tool access and usage budgets as HybridAgent.

*ReAct* (Yao et al., 2023): ReAct interleaves natural-language reasoning with actions and observations from an external environment. This interaction loop allows an agent to gather information, revise its reasoning, and determine subsequent actions based on tool feedback.

*Search-o1* (Li et al., 2025a): Search-o1 augments large reasoning models with adaptive search to address knowledge gaps encountered during reasoning. Its Reason-in-Documents module analyzes retrieved documents and extracts relevant evidence before incorporating it into ongoing reasoning.

Table 10: Hyperparameter Setups of HybridSFT.

| Hyperparameter | Value |
|---|---|
| Base model | Qwen3-4B/8B/14B-Base |
| Maximum sequence length | 16,888 |
| Effective batch size | 16 |
| Backbone learning rate | $7 \times 10^{-6}$ |
| HybridLink learning rate (Activation / Self-conditioning) | $2 \times 10^{-4}$ / $5 \times 10^{-5}$ |
| Weight decay | 0.0 |
| Warmup ratio | 0.1 |
| Learning rate scheduler | Cosine |
| Latent regression weights ($\lambda_{\text{cos}}$ / $\lambda_{\text{mse}}$) | 1.0 / 0.2 |
| Number of training epochs | 3 (1 / 1 / 1) |

*WebThinker* (Li et al., 2025b): WebThinker equips LLMs with a Deep Web Explorer that supports iterative search, webpage navigation, and information extraction. Its Think-Search-and-Draft strategy interleaves reasoning, evidence collection, and report generation to support complex research tasks.

Overall, these baseline methods provide complementary comparisons across training stages, text-based and latent-space reasoning, and tool-integrated agentic settings.

### H.3 Implementation Details

**Models and Training Details.** We instantiate HybridAgent with Qwen3-Base backbones (Yang et al., 2025) at three scales of 4B, 8B, and 14B. For HybridLink, we use the same two-layer design across all three backbones. The HybridSFT training data is constructed by converting 54K text-only sources (Dong et al., 2025a;b) into hybrid trajectories. We apply format and HybridLink activation for one epoch each, followed by one additional epoch of self-conditioned training. HybridRL training utilizes 12K deep reasoning examples (Yu et al., 2025) for math and 1K agentic reasoning and search examples (Dong et al., 2025b) for deep search. We train HybridAgent with the GRPO objective and set rollout batches of 512 trajectories ($64 \times 8$), followed by one optimization pass over 8 minibatches.

**Training Hyperparameters.** Table 10, 11, and 12 provide detailed parameter setups on HybridSFT and HybridRL. Following Dong et al. (2025b); Yu et al. (2025), we use HybridAgent checkpoints from deep reasoning training for mathematical evaluations and the checkpoint from agentic search training for knowledge-intensive search and agentic reasoning evaluations, as detailed in Appendix H.1.

**Inference Setups.** During inference, we set the temperature, top-p, and top-k to 0.6, 0.95, and 20, respectively. The generation budget is adjusted for each task based on its relative difficulty and counts both text tokens and latent steps, with limits of 4,096 for mathematical reasoning and knowledge-intensive search tasks, and 24,000 for challenging agentic reasoning tasks. For each latent span, we enforce a minimum of 16 and a maximum of 256 steps to ensure sustained and semantically rich latent reasoning. HybridAgent interacts with external environments primarily through two types of tools: a Python interpreter within sandbox environments and a live search engine (e.g., Tavily). We cap tool usage at 10 calls per trajectory and retrieve the top 10 snippets for each search query. We utilize GPT-5.6-Luna as both the search-page summarizer and LLM judge. All training and inference are conducted on 32 H100 GPUs. We also provide detailed prompt templates in Appendix I.

**Infrastructure and Configurations.** As existing training and inference frameworks such as verl (Sheng et al., 2025) and vLLM (Kwon et al., 2023) do not directly support multi-turn hybrid generation, we extend their infrastructure to enable hybrid training and generation in HybridAgent. A shared generation loop coordinates text mode, latent mode, and tool interaction while maintaining each trajectory's persistent KV cache as continuous working memory across turns. The agent enters latent mode through the special entry anchor `<latent>`, feeds HybridLink-generated latent inputs back into the backbone, and returns to text mode when the nearest-token readout matches the exit anchor `</latent>`, as in Eq. 5. Tool observations are tokenized and processed against the existing working memory, appending new KV entries without reconstructing or re-prefilling earlier context.

The engine also supports batched rollouts with per-sequence state tracking, allowing trajectories to occupy different generation states within the same batch. Pending tool calls execute concurrently, with waiting and completed sequences masked from further generation. Per-sequence attention masks and position tracking preserve each trajectory's causal context as generation proceeds. For HybridRL,

Table 11: Hyperparameter Setups of HybridRL on Agentic Search.

| **Hyperparameter** | **Value** |
|---|---|
| Initialization | 8B/14B HybridSFT checkpoint |
| Unique prompts per rollout batch | 64 |
| Rollouts per prompt (group size) | 8 |
| Minibatches per rollout batch | 8 |
| Optimization epochs per rollout batch | 1 |
| Totoal rollout batch size | 512 trajectories ($64 \times 8$) |
| Optimization per rollout batch | 1 pass over 8 minibatches |
| Rollout sampling temperature | 1.0 |
| Rollout sampling top-p | 1.0 |
| Reward $r_{\text{correct}}$ | token-level F1 $\in [0, 1]$ |
| Reward $r_{\text{format}}$ | 0 / -1 |
| Reward $r_{\text{tool}}$ | 0.1 |
| Latent steps per latent block (min/max) | 16 / 256 |
| Max tool calls per rollout | 10 |
| Weight decay | 0.01 |
| KL penalty coefficient | 0 |
| Learning rate schedule | constant |
| Training steps | 150 |
| Maximum sequence length | 16,888 |
| Backbone learning rate | $1 \times 10^{-6}$ |
| Optimizer | AdamW, $\beta = (0.9,\ 0.95)$ |
| Search API Setup (per call) | top-10, 1000 characters each |

Table 12: Hyperparameter Setups of HybridRL on Deep Reasoning.

| **Hyperparameter** | **Value** |
|---|---|
| Initialization | 4B/8B/14B HybridSFT checkpoint |
| Unique prompts per rollout batch | 64 |
| Rollouts per prompt (group size) | 8 |
| Minibatches per rollout batch | 8 |
| Optimization epochs per rollout batch | 1 |
| Total rollout batch size | 512 trajectories ($64 \times 8$) |
| Optimization per rollout batch | 1 pass over 8 minibatches |
| Rollout sampling temperature | 1.0 |
| Rollout sampling top-$p$ | 1.0 |
| Reward $r_{\text{correct}}$ | Math Verify (0 / 1) |
| Reward $r_{\text{format}}$ | $0$ / $-1$ |
| Reward $r_{\text{tool}}$ | $+0.1$ |
| Latent steps per latent span (min/max) | 16 / 256 |
| Max tool calls per rollout | 5 |
| Weight decay | 0.01 |
| KL penalty coefficient | 0 |
| Learning rate schedule | Constant |
| Training steps | 200 |
| Maximum sequence length | 16,888 |
| Backbone learning rate | $1 \times 10^{-6}$ |
| Optimizer | AdamW, $\beta = (0.9,\ 0.95)$ |

exact latent-context replay recomputes sampled text-token likelihoods in a single differentiable forward pass while keeping recorded latent inputs unchanged. We check rollout–replay agreement of log-probabilities at unchanged policy parameters. Multi-node data parallelism, sharded optimizer states, mixed precision, gradient checkpointing, and resumable checkpoints support full-parameter backbone optimization. Overall, the same engine supports RL rollouts, training-time validation, and downstream evaluation, ensuring consistent mode transitions and working-memory handling across configurable sampling settings, generation budgets, and tool-call setups.

# I PROMPT TEMPLATE

**Prompt Template for HybridAgent Training and Inference**

**System Prompt** *(shared across examples)*

```
You are a helpful assistant that solves the given question step by step with
the help of an online live search tool and a python interpreter tool. Given a
question, you first think about the reasoning process and then provide the
final answer, invoking the search tool for factual information and the python
interpreter tool for calculations whenever needed.

Format:
- <think> ... </think>: your reasoning. Start every round with <think>; close
it with </think> and then output exactly one of: a <search> call, a <python>
call, or the final <answer>.

- <latent> ... </latent>: latent reasoning, allowed only inside <think> ...
</think>. Put routine internal reasoning (planning, recalling, restating,
routine double-checking) in latent mode. Outside <latent>, write only the few
decisive steps - key facts, decisions, and conclusions - as plain text.
Always start and end the think block with plain text: open with a brief
plain-text step (e.g. what the question or the latest result tells you), and
close with the plain-text decision or conclusion that leads to your next
action.

- <search> query </search>: call the online live search tool.

- <python> code </python>: call the python interpreter tool; use print() to
show the output.

- <result> ... </result>: the tool output, inserted by the system after your
</search> or </python>. Never write <result> yourself. After each result,
start the next round with <think> - or, if the result directly settles the
question, give the final <answer> immediately. If a result is empty, an
error, or a tool-limit notice, do not repeat the same call - adjust your
approach or give your best final answer.

- <answer> ... </answer>: the final response, with the exact final answer
enclosed in \boxed{} in latex format.

Format example:
<think>
This is a key first step.<latent>This is the latent reasoning
process.</latent>This is a key reasoning step.
</think><search>search query here</search><result>search result
here</result><think>
This is a key first step.<latent>This is the latent reasoning
process.</latent>This is the key decision step.
</think><python>python code here</python><result>python interpreter result
here</result><think>
This is a key first step.<latent>This is the latent reasoning
process.</latent>This is the key conclusion.
</think><answer>\boxed{answer here}</answer>
```

**User Prompt** *(question for the current example)*

```
{question}
```

**Prompt Template for Downstream LLM Judge**

```
Given a Question and its Golden Answer, verify whether the Predicted Answer
is correct. The prediction is correct if it fully aligns with the meaning and
key information of the Golden Answer. Respond with "Correct" if the
prediction is correct and "Incorrect" otherwise.

Question: {question}
Golden Answer: {labeled_answer}
Predicted Answer: {pred_answer}
```

**Prompt Template for LLM-Guided Hybrid Data Annotation**

**System Prompt**

```
You are a data expert for classifying Chain-of-Thought (CoT) steps produced by
an assistant that can use tools (search, python) and outputs a final answer.

For each CoT step you will be given, you must label it as either:
- **0 (latent mode)**: Internal thinking that is not critical for auditing or
debugging. Prefer latent for routine reasoning, filler, or steps that do not
need to be shown as text for correctness or interpretability.
- **1 (text mode)**: Must be text when (a) the step clearly calls an external
tool or relates to a tool call (e.g. contains <search>...</search>,
<python>...</python>, <result>...</result>, or <answer>...</answer>), or (b)
the step is important for auditing/debugging (e.g. the step that states the
final answer, or a core middle step that corresponds to the key solution
decision). For efficiency, do NOT mark too many steps as text: only those that
are clearly tool-related or genuinely important for understanding the solution.

Balance: Prefer latent for non-essential inner reasoning; use text only for
tool steps and a small set of critical reasoning/answer steps.

When you respond, you MUST first write your reasoning, then output the final
JSON array inside a ```json ... ``` code block. Do not skip the reasoning
section.
```

**User Prompt**

```
Below are {num_steps} chain-of-thought steps from one conversation, labeled
step0 to step{N}.

{steps_block}

Do the following in order:
1. Reasoning: for each step from step0 to step{N}, give its label (0 = latent,
1 = text) and a brief reason, e.g. "step0: 0 - internal planning"; "step3: 1 -
contains <search>".
2. Check: confirm that every step is labeled and that your array will have
exactly {num_steps} entries.
3. Final answer: end with a JSON code block containing only the array of
{num_steps} integers (0 or 1), in order from step0 to step{N}:

An example of the final array format for the given steps is:

```json
[0,1,1,0]
```
```

# J CASE STUDY OF HYBRIDAGENT

**HybridAgent on Mathematical Reasoning** — MATH500

**QUESTION** If each point of the circle $x^2 + y^2 = 25$ is reflected in the point $(4, 1)$, the set of image points satisfies the equation $x^2 + ay^2 + bx + cy + d = 0$. Compute ordered quadruple $(a, b, c, d)$ of real numbers.

**TURN 01**

**TEXT** To solve this problem, we need to find the image of each point $(x_1, y_1)$ on the circle $x^2 + y^2 = 25$ when it is reflected in the point $(4, 1)$.

**LATENT** The reflection of a point $(x_1, y_1)$ in another point $(x_0, y_0)$ is given by the formula:
$(x_2, y_2) = (2x_0 - x_1, 2y_0 - y_1)$.
In this case, $(x_0, y_0) = (4, 1)$, so the reflection of $(x_1, y_1)$ is:
$(x_2, y_2) = (2 \cdot 4 - x_1, 2 \cdot 1 - y_1) = (8 - x_1, 2 - y_1)$.

**TEXT** Now, we need to find the equation of the locus of these image points $(x_2, y_2)$.

**LATENT** Since $(x_1, y_1)$ lies on the circle $x^2 + y^2 = 25$, we substitute $x_1 = 8 - x_2$ and $y_1 = 2 - y_2$ into the circle's equation: $(8 - x_2)^2 + (2 - y_2)^2 = 25$.

**TEXT** We will expand and simplify this equation to find the values of $a$, $b$, $c$, and $d$. Let's start by expanding the squares:

$(8 - x_2)^2 = 64 - 16x_2 + x_2^2$,

$(2 - y_2)^2 = 4 - 4y_2 + y_2^2$.

Adding these together, we get: $64 - 16x_2 + x_2^2 + 4 - 4y_2 + y_2^2 = 25$.

**LATENT** Simplify the left side: $x_2^2 + y_2^2 - 16x_2 - 4y_2 + 68 = 25$. Rearrange to standard form by moving 25 to the left side:

$x_2^2 + y_2^2 - 16x_2 - 4y_2 + 43 = 0$.

Now we can identify the coefficients $a$, $b$, $c$, and $d$: $a = 1, \quad b = -16, \quad c = -4, \quad d = 43$.

**TEXT** Let's confirm this by implementing the steps in Python to ensure the result is accurate.

**TOOL** python

```
from sympy import symbols, Eq, simplify
x1, y1, x2, y2 = symbols('x1 y1 x2 y2')
...
polynomial_form = Eq(simplified_eq.lhs - simplified_eq.rhs, 0)
print(polynomial_form)
```

**ERROR**

```
Python exited with code 1
Partial stderr:  Traceback (most recent call last):
  File "<string>", line 53, in <module>
  File "<string>", line 38, in _execute_user_code
  File "<python-tool>", line 12, in <module>
NameError:  name 'solve' is not defined
```

**TURN 02**

**TEXT** It seems that the symbolic computation code had an issue because the `solve` function was not imported. Let me correct that by importing the necessary functions from `sympy`. I'll also ensure the code is properly encapsulated within the environment.

**TOOL** python

```
from sympy import symbols, Eq, solve, simplify
x1, y1, x2, y2 = symbols('x1 y1 x2 y2')
...
print(polynomial_form)
```

**RESULT**

```
Eq((x2 - 8)**2 + (y2 - 2)**2 - 25, 0)
```

**TURN 03**

**LATENT** The output confirms our earlier manual calculation. The equation of the locus of the image points is: $(x_2 - 8)^2 + (y_2 - 2)^2 - 25 = 0$. Expanding this, we get: $x_2^2 - 16x_2 + 64 + y_2^2 - 4y_2 + 4 - 25 = 0$. Simplifying further: $x_2^2 + y_2^2 - 16x_2 - 4y_2 + 43 = 0$.

**TEXT** Thus, the values of $a$, $b$, $c$, and $d$ are: $a = 1, \quad b = -16, \quad c = -4, \quad d = 43$.
So, the ordered quadruple $(a, b, c, d)$ is:

**ANSWER** $\boxed{(1, -16, -4, 43)}$ ✓ Correct answer

**HybridAgent on Knowledge-Intensive Reasoning** — MuSiQue

**QUESTION** What field of work was the person who discovered the mathematical relationship between peaks and curves of light at different temperatures in?

**TURN 01**

**TEXT** Okay, so I need to figure out what field of work the person was who discovered the mathematical relationship between peaks and curves of light at different temperatures. Hmm, let me think about this step by step.

**LATENT** First, I remember that there's a specific in in physics or to the color of light by objects at different temperature. Maybe it's some like the black's function or Wait, or, that's more about the energy of energy from a black body.

**TEXT** Wait, the question mentions "peaks and curves of light at different temperatures." That sounds like the relationship between temperature and the spectrum of emitted light. Oh, right! The person I'm thinking of is Max Planck. He's known for his work on black-body radiation. Planck's law describes the spectral density of electromagnetic radiation emitted by a black body in thermal equilibrium at a given temperature.

**LATENT** So the question is about for the field of work. So I need was a physics, but I need to be more. Let me do a quick search to confirm. I'll look up the discovery related to peaks and curves of light at different temperatures and the corresponding field.

**TOOL** search `discovery of mathematical relationship between peaks and curves of light at different temperatures field`

**RESULT** **Page 1:** Wien's displacement law states that the black-body radiation curve for different temperatures will peak at different wavelengths that are inversely

**Page 2:** Each blackbody curve represents a blackbody at a specific temperature, and each curve peaks at a different wavelength, so you can figure out the temperature of a blackbody by measuring the wavelength at which its [. . . ]

**TURN 02**

**TEXT** Okay, so the question is asking about the field of work of the person who discovered the mathematical relationship between peaks and curves of light at different temperatures.

**LATENT** From the search results, I can see that the key is to the's distance law. The is about by William Wien. The search results mention that Wien's displacement law was discovered by Wilhelm Wien. Now I need to find out what field of work Wilhelm Wien was in. Let me do another search to find out William Wien's field of work.

**TEXT** Wait, I already have the answer from the first search. The question is about the field of work. Since the discovery is related to black-body radiation and physics, it's safe to assume that Wilhelm Wien was a physicist.

**LATENT** Let me verify this through another search query:

**TOOL** search `Wilhelm Wien field of work`

**RESULT** **Page 1:** Wilhelm Wien From ETHW Jump to:navigation, search Wilhelm Wien Birthdate : 1864 Birthplace : East Prussia Associated organizations : University of Munich Fields of study : Quantum physics Awards : Nobel Prize in physics Biography Wilhelm Wien was a German physicist and Nobel-Prize winner who pioneered the study of quantum physics. [...] From early in his career, Wien [. . . ]

**TURN 03**

**TEXT** Okay, so the question is asking about the field of work of the person who discovered the mathematical relationship between peaks and curves of light at different temperatures.

**LATENT** From the first search, I found that the person is related to William's distance law, which was found by William Wien. Then, I did another search to find Wilhelm Wien's field of work. The search results clearly state that Wilhelm Wien was a German physicist.

**TEXT** Multiple sources confirm that his primary area of work was physics, specifically in the study of thermal radiation and quantum physics. He was awarded the Nobel Prize in Physics for his work on black-body radiation.

**LATENT** Then the field of work is physics. The person who discovered the mathematical relationship between peaks and curves of light at different temperatures was Wilhelm Wien, and his field of work was physics.

**ANSWER** Physics ✓ Correct answer

**HybridAgent on Agentic Reasoning** GAIA

**QUESTION** Use the density values from Marisa Alviar-Agnew and Henry Agnew's CK-12-licensed *Introductory Chemistry* materials on LibreTexts, as compiled on 08/21/2023.

I have one gallon of honey and one gallon of mayonnaise at 25°C. If I remove one cup of honey at a time, how many cups must I remove for the remaining honey to weigh less than the mayonnaise? Assume that the containers have equal mass.

**TURN 01**

**TEXT** I need the smallest number of one-cup removals that makes the remaining honey lighter than one gallon of mayonnaise.

**LATENT** Because the containers have equal mass, I only need to compare their contents. I first need the densities of honey and mayonnaise at 25°C from the specified chemistry reference.

**TEXT** Since the containers have equal mass, I can compare their contents using $m = \rho V$. I'll first look up the densities at 25°C.

**TOOL** search
```
density of honey and mayonnaise in g/mL
```

**RESULT** Page 1: Plug in the values we measured earlier, and we'll get the densities of honey and mayonnaise. The table above shows the densities of honey and mayonnaise. We can see that both substances have a density of approximately 5g/mL. Page 2: Table \(\PageIndex{1}\): Densities of Common Substances | Substance | Density at 25°C (g/cm3) | --- | | blood | 1.035 | | body fat | 0.918 [...]

**TURN 02**

**TEXT** For the calculation, I use $1.42\,\mathrm{g/mL}$ for honey and $0.91\,\mathrm{g/mL}$ for mayonnaise. I'll convert the volumes to milliliters to calculate their masses.

**LATENT** Using U.S. liquid measures, one gallon is approximately $3785.41\,\mathrm{mL}$ and one cup is approximately $236.588\,\mathrm{mL}$. The initial mass of the honey is

$m_{\text{honey}} = 1.42 \times 3785.41 = 5375.2822\,\mathrm{g}.$

**TEXT** The honey initially weighs about $5375.28\,\mathrm{g}$, while the mayonnaise weighs $0.91 \times 3785.41 \approx 3444.72\,\mathrm{g}$. I now need the mass removed per cup to determine when the honey falls below this value.

**LATENT** Each cup of honey has mass

$m_{\text{cup}} = 1.42 \times 236.588 = 335.95496\,\mathrm{g}.$

After removing $n$ cups, the required inequality is

$5375.2822 - 335.95496n < 3444.7231,$

$$n > \frac{5375.2822 - 3444.7231}{335.95496} \approx 5.74648.$$

The smallest valid integer is $n = 6$. I'll verify this with Python.

**TEXT** Each cup removes $335.95496\,\mathrm{g}$. The condition $5375.2822 - 335.95496n < 3444.7231$ requires $n > 5.74648$. I'll check in Python that six cups are sufficient and five are not.

**TOOL** python
```
from math import floor
rho_h, rho_m = 1.42, 0.91
gallon_ml, cup_ml = 3785.41, 236.588
m_h = rho_h * gallon_ml
m_m = rho_m * gallon_ml
m_cup = rho_h * cup_ml
n = floor((m_h - m_m) / m_cup) + 1
assert m_h - n * m_cup < m_m
assert m_h - (n - 1) * m_cup >= m_m
print(f"Number of cups to remove:  {n}")
```

**RESULT**
```
Number of cups to remove:  6
```

**TURN 03**

**TEXT** Python confirms the minimum: six cups leave about $3359.55\,\mathrm{g}$ of honey, below the mayonnaise's $3444.72\,\mathrm{g}$; five cups leave about $3695.51\,\mathrm{g}$, which is still too heavy.

**LATENT** Six one-cup removals are therefore necessary and sufficient.

**ANSWER** $\boxed{6}$ ✓ Correct answer