AutoTool: Automatic Scaling of Tool-Use Capabilities in RL via Decoupled Entropy Constraints

Anonymous authors
Paper under double-blind review

Abstract

Tool use represents a critical capability for AI agents, with recent advances focusing on leveraging reinforcement learning (RL) to scale up the explicit reasoning process to achieve better performance. However, there are some key challenges for tool use in current RL-based scaling approaches: (a) direct RL training often struggles to scale up thinking length sufficiently to solve complex problems, and (b) scaled-up models tend to overthink simpler problems, resulting in substantial token inefficiency. To address these challenges, we propose a novel training paradigm that first employs warm-up supervised fine-tuning to help models distinguish between simple and complex problems, followed by RL that enable models to automatically determine appropriate reasoning trajectories. Furthermore, to tackle the issue of automatic thinking-length scaling, we discover that entropy-based optimization objectives effectively maintain model diversity while successfully unlocking the model's scaling capabilities. Based on this insight, we introduce an entropy-based long-short reasoning fusion RL strategy. Our experiments on three benchmarks demonstrate that model successfully achieves auto-scaling for efficient tool use, achieving significant 9.8% accuracy improvements while reducing computational overhead by ~81%.

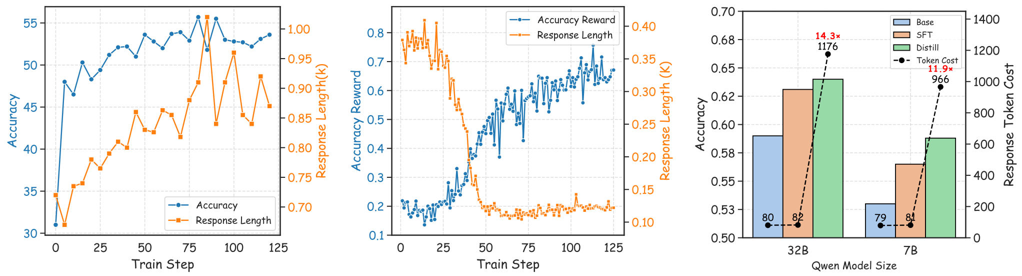
1 Introduction

Integrating agentic large language models (LLMs) with external tools has emerged as a pivotal advancement, and has become a defining feature of advanced agentic models (OpenAI, 2025; K2, 2024). It significantly enhances a model's ability to address complex tasks (Qu et al., 2025; Wang et al., 2024), and opens up many practical uses across different fields. For example, it supports the automation of reasoning tasks (Jin et al., 2025; Li et al., 2025b), and enables agent applications (Zihan et al., 2025; Ouyang et al., 2025). Therefore, research on agentic tool use represents a critical pathway toward artificial general intelligence. In this task, models respond to queries by dynamically selecting and invoking relevant tools from an available pool.

Test-time scaling (TTS) is a approach to language modeling that uses extra test-time compute to improve performance (Muennighoff et al., 2025). Currently, scaling up a model's explicit reasoning length via Reinforcement Learning with Verifiable Rewards (RLVR) is a effective to achieve TTS. Compared to the prevalent Supervised Fine-Tuning (SFT) approach, which imitates reasoning patterns from labeled high-quality examples to teach models using tools (Liu et al., 2024; Zhang et al., 2024a), RLVR better fosters intrinsic reasoning, rather than making models memorize training trajectories (Chen et al., 2025). It has demonstrated robustness in mathematics (Shao et al., 2024) and coding (Pan & Liu, 2025), while also driving a paradigm shift from SFT to RL in LLM training. Thus, exploring suitable scaling strategies is critical to advance effective agentic tool-use.

To this end, we pre-analyze the training paradigm for TTS in tool use, as shown in Figure 1. Under direct RL training, contrary to mathematical tasks where response length scales with improving accuracy, we observe that models suffer from *reasoning collapse* in tool use, a phenomenon where models fail to sufficiently extend thinking[1] length to solve complex problems. More importantly, we

[1] In this paper, the terms *thinking* and *long reasoning* are used interchangeably, both referring to responses that contain an explicit reasoning process.



(a) direct RL training in Math

(b) direct RL training in tool use

(c) Cost in scaled-up models

Figure 1: The training paradigms for TTS in tool-use: (a) direct RL enables scaling up response length as accuracy improves in mathematical tasks; but (b) it fails to scale in tool-use tasks, where reasoning collapses into short trajectories; (c) scaled-up models (e.g., distillation models) incur significant token costs, as they require lengthy reasoning trajectories for all queries.

find many tool-use problems can be solved with short reasoning trajectories, yet scaled-up models generate excessively long trajectories that cause unnecessary resource consumption. Therefore, an adaptive model that dynamically integrates short and long reasoning is highly desirable.

More analysis in Section 2 reveal that low entropy, which quantifies a model's response certainty and exploration capability, leads to insufficient reasoning length, limiting the robustness of LLMs in tackling hard problems. These findings motivate our proposed method, which decouples long and short reasoning to prevent dominance interference while incorporating entropy constraints to enable long-trajectory reasoning. Therefore, we propose a decoupled adaptive entropy constraint strategy for RL. It first performs warm-up using a constructed mixed dataset of long and short reasoning trajectories to perceive data difficulty. The strategy decouples the policy loss between short and long reasoning, then applies varying entropy constraint strengths to regulate the thinking mode while maintaining concise responses for simple problems. This enables differentiated exploration control across reasoning modes, with adjusted entropy strength set above a target entropy in long reasoning to preserve exploration capacity.

Experiments on three benchmarks show: (1) Our ~7B model leads at comparable size models (e.g., +11.95% compare to SFT-model). (2) Beyond performance, our auto-scaling model boosts accuracy by 9.8% compared to the distilled model and cuts inference token cost by ~81%. Notably, our model's thinking rate reaches 45% in complex scenarios but 0% in simple ones. Moreover, visualizations of the training process confirm that our approach generates concise responses for simple cases while extending reasoning trajectories by 5× for complex questions. This contrast demonstrates that the model has learned to automatically adjust test-time scales based on sample difficulty, ultimately supporting improved inference efficiency.

2 Preliminary Study

In this section, we present extensive experiments to highlight the challenges of achieving test-time scaling for agentic tool use, and thereby motivate our method.

2.1 Task Overview

In agentic tool use, the LLM receives a user query q along with candidate tools, represented as $\mathcal{T} = \{t_0, t_1, \ldots, t_{|\mathcal{T}|}\}$. The purpose of LLM is to fulfill the user's intent by executing a specific sequence of tools. We formalize this decision-making process as $y \sim \pi(y \mid q, s, \mathcal{T})$, where $\pi(\cdot)$ represents the policy model, s denotes the task state (e.g., historical context), and y represents the actions taken by the model, such as selecting or executing tool calls. A review of related work is provided in Appendix A.

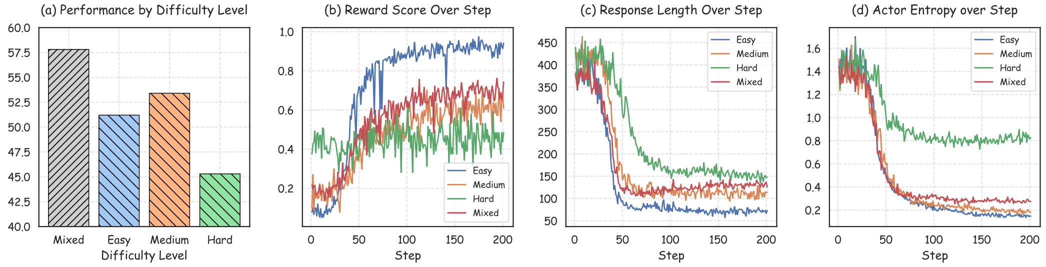


Figure 2: Impact of difficulty distributions. *Easy* and *Medium* converged successfully, while *Hard* failed (a, b). However, collapse occurred across all three subsets (c), with the same trend observed in entropy (d). This indicates that data distribution has no correlation with collapse, whereas low entropy exhibits a strong positive correlation.

2.2 Training Paradigms Analysis

We analyze training paradigms for scaling reasoning process in tool use, including RL training and SFT with distillation, using Qwen2.5-series models to conduct training on the public *ToolACE* dataset (Liu et al., 2024) and evaluation via *BFCL* (Yan et al., 2024) (details in Section 4.1).

(1) For direct RL, we applied RL (specifically GRPO (Shao et al., 2024)). As shown in Figure 3b, we observe a divergence between the model's performance and response length: as training steps increased, performance improved while response length decreased sharply. This indicates the reasoning patterns collapsed into short reasoning trajectories, failing to scale-up in test-time. This result contradicts widely accepted findings from training on complex reasoning tasks (e.g., mathematics) (Hugging Face, 2025; Zeng et al., 2025b), as shown in Figure 3a, where we adopt experimental results from Zeng et al. (2025a). The evaluation results presented in Table 1 indicate that performance in complex tool-use scenarios (e.g., Multi-Turn) decreased noticeably compared to the distilled SFT model. This indicates that the reasoning collapse phenomenon limits the model's robust performance on complex problems.

(2) For SFT with distillation, we conducted base SFT and distillation from reasoning LLM (DeepSeek-AI, 2025b), respectively. As shown in Figure 1c, distilled models showed no noticeable accuracy gain over the base SFT, but increased output token costs by more than 10×. This suggests that many agentic tool-use problems can be solved with short reasoning trajectory while excessive long trajectory leads to unnecessary resource consumption.

2.3 Pre-study on Reasoning Pattern Collapse.

To investigate the causes of reasoning pattern collapse, we conducted an in-depth analysis of data difficulty distribution and information entropy.

2.3.1 Data Distribution

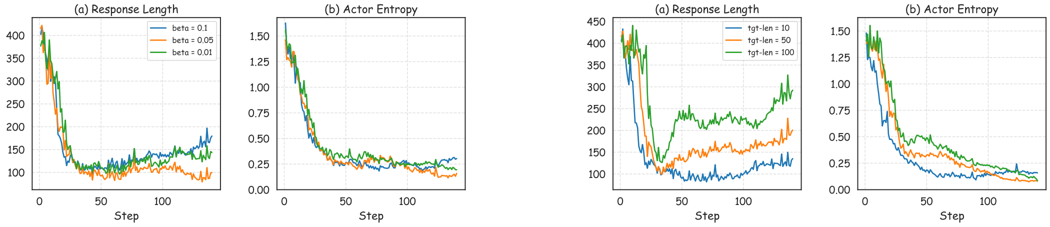
Intuitively, we hypothesized that the sample difficulty distribution might exert a critical influence. Guided by this hypothesis, we used a base model to perform 8 rounds of reasoning on the training data and calculated pass@8. The resulting distribution (shown in Appendix B) reveals that **easy** samples (with $8/8$ correct inferences) and **hard** samples (with $0/8$ correct inferences) accounted for 47% and 31.8% of the dataset, respectively, while **medium** samples made up a smaller proportion.

We then conducted separate RL training runs on these three subsets, evaluated the resulting models, and reported their training dynamics in Figure 2. Notably, reasoning pattern collapse persisted across all three subsets: after an initial exploration phase, easy samples led to rapid convergence, intermediate samples resulted in slower convergence, and hard samples showed no convergence. These findings indicate that the sample difficulty distribution can slightly reduce convergence speed, but no correlation with collapse has been observed.

The actor model's information entropy quantifies its exploration capability during training. As shown in Figure 2d, entropy decreases rapidly, with dynamics closely aligning with reasoning pat-

Table 1: Evaluation results on the BFCL benchmark, which includes three sub-metrics: Non-live, Live, and Multi-Turn (including multi-turn and long-context tool-use scenarios). l denotes target response length in response constraint, and β denotes coefficients of entropy constraint.

Model	Think?	Non-Live	Live	Multi-Turn	Overall Acc
Base LLM	✗	86.46	67.44	7.62	53.69
w/ SFT	✗	86.65	75.11	6.75	56.90
w/ distilled SFT	✓	87.35	79.59	16.95	59.23
w/ GRPO	✓	87.06	78.22	8.38$_{\downarrow 8.57}$	57.81
w/ length constraint					
$+l = 100$	✓	87.30	71.23	7.92	55.37
$+l = 50$	✓	87.76	78.43	8.78	58.12
$+l = 10$	✓	89.76	77.33	8.89	58.27
w/ entropy constraint					
$+\beta = 1e{-}2$	✓	87.47	79.13	9.48	59.33
$+\beta = 5e{-}2$	✓	87.21	77.96	10.02	58.91
$+\beta = 1e{-}1$	✓	88.32	80.42	15.86$_{\uparrow 7.48}$	61.86



(a) RL with entropy constraint

(b) RL with length penalty

Figure 3: Training dynamics visualized for entropy constraint (a) and length penalty (b). The entropy constraint partially increases response length, yet the length penalty does not mitigate low entropy.

tern collapse. Additionally, comparing three subsets, the final converged entropy increases sequentially from simple to complex samples. This reveals: simple problems elicit high certainty in short reasoning (perceiving extended exploration risks suboptimal solutions); complex problems face inherent challenges, with short reasoning advantage dominance further discouraging exploration and driving default to brief responses. This finding demonstrates a strong positive correlation between low entropy and collapse.

2.3.2 Information Entropy Constraints

To explore reasoning collapse-entropy connections, we incorporated entropy constraints into the policy loss function. Inspired by He et al. (2025), we designed a mechanism to maintain the entropy (e) at a reasonable level throughout training. The entropy loss is defined as:

$$loss_k^e = \beta \cdot \mathbb{I}\{e_k \leq \texttt{tgt-ent}\} \tag{1}$$

where k is the training step, β is the coefficient, and we set $\texttt{tgt-ent}$=0.1. Notably, the entropy loss is only activated when $e_k \leq \texttt{tgt-ent}$, ensuring the model's entropy remains lower-bounded by the target value. For comparative purposes, we also implemented a short-response penalty by configuring the reward function to penalize response below the target length l. The evaluation results (presented in Table 1) show that the length constraint did not improve the model's test performance. In contrast, the entropy constraint yielded partial performance gains (visualizing the training process in Figure 3). However, the effectiveness is highly sensitive to β: in multi-turn, the maximum positive gain was achieved when $\beta = 1e - 1$, whereas results for other β are comparable to *w/ length constraints*. This sensitivity highlights the difficulty of pre-selecting an optimal entropy coefficient, indicating that dynamically adjusting β during training is necessary.

Therefore, we propose a novel strategy: decoupled adaptive entropy constraints. It decouples entropy constraint for short and long reasoning and adaptively tunes the entropy coefficient, addressing both the collapse caused by low entropy and the sensitivity of static coefficients.

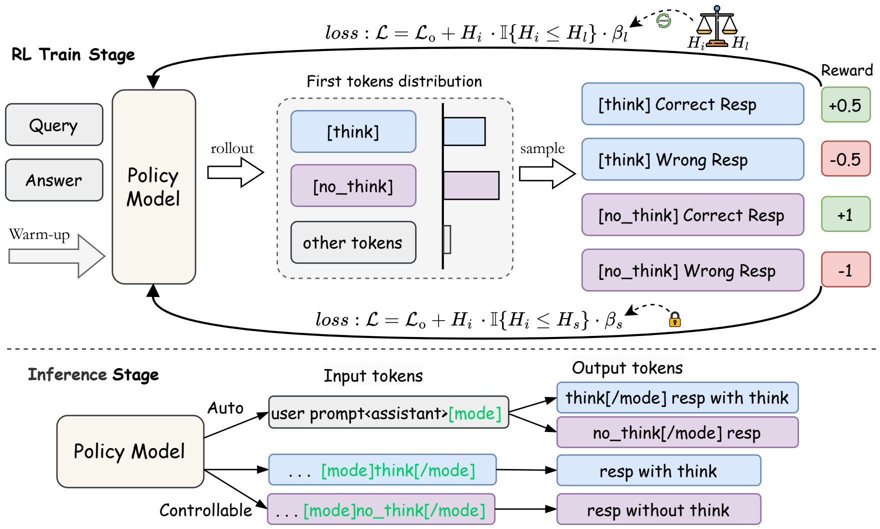


Figure 4: The overview of decoupled adaptive entropy constraint. It achieves automatic scaling by decoupling different reasoning modes through the application of differentiated entropy constraints. Adaptive entropy constraint strength for long reasoning. During the inference, the model can automatically or controllably switch inference modes by pre-pending a response prefix in Input tokens.

3 Methodology

Our method first performs warm-up SFT to perceive sample difficulty (details in 3.1), followed by RLVR training with decoupled adaptive entropy constraints, as shown in Figure 4.

3.1 Data Preparation and Warm-up

To support robust general-purpose tool use via RL, we constructed a mixed dataset covering diverse tool-use scenarios from public sources: ToolACE (Liu et al., 2024), xLAM (Zhang et al., 2024a; Prabhakar et al., 2025), Hermes Function-Calling (interstellarninja, 2024). More details are provided in Appendix B. To create a balanced dataset encompassing diverse complexity levels and tool usage scenarios, we randomly downsampled the raw data. Moreover, we adopted the following strategies to develop a Public agentic Tool-use dataset (**PubTool**), as presented in Table 2.

Warm-up Training. To help the model initially perceive data difficulty, we propose SFT for warm-up training by mixing long and short reasoning data. To construct such mixed thinking data, we performed multiple inferences (calculating pass@8) on the training data using Qwen2.5-7B-Instruct (no-thinking model) and Qwen3-32B (thinking model), respectively. For each response turn, we adopted the ground truth as the label if the no-thinking model's output was correct; otherwise, we adopted the thinking model's answer with explicit long reasoning if it was correct. More Details of data preparation are shown in Appendix §B. We design an auto-thinking template (details in Appendix §F) to enable the model to select reasoning modes based on data difficulty. Finally, we conducted SFT on the base model for warm-up, preparing for subsequent RL scaling.

Quality Refinement for RL data. To efficiently support auto-scaling RL training, we employed the following data enhancement strategies: First, from data distribution analysis in Section §2.3.1, we observed that the original dataset was dominated by overly simple and excessively difficult samples. Overly simple samples offer limited value for RL training and lack generalization, while overly difficult samples either exceed model capabilities or contain noise. We therefore randomly removed half of both simple and difficult samples to balance the dataset distribution. Additionally, inspired by Li et al. (2025a), we prioritized training samples based on their alignment with model learning trajectories. Specifically, we performed multi-epoch GRPO training on all training data, computed changes in their reward scores, and calculated each sample's variance relative to the average reward.

Table 2: Data statistics of **PubTool** in data collection and construction. Subscript text in the SFT data table indicates the thinking rate in all turns.

	ToolACE	xLAM	Hermes Function-Calling
Raw Data	11.3k	65k	7.1k
Downsampled	11.3k	15k	7.1k
	PubTool		
Processed	SFT data 8.2k$_{(9.2\%)}$		RL data 7k

Lower variance indicated higher alignment. Through these processes, we downsampled the RL dataset from 21k to 7k samples. For a detailed analysis of its effects, please refer to Appendix B.

3.2 Decoupled Adaptive Entropy Constraints

To enable automatic scaling in agentic tool use, we propose a *decoupled adaptive entropy constraints* strategy for RLVR. The objective policy loss integrates the surrogate objective from native RLVR (e.g., GRPO) with a mechanism that: (1) decouples entropy regulation between short and long trajectories; (2) adaptively adjusts the entropy strength in long reasoning trajectories to preserve exploration capacity.

Specifically, let π_θ be the policy, $H_i = -\mathbb{E}_{a\sim\pi_\theta(\cdot|s_i)}[\log \pi_\theta(a|s_i)]$ is the entropy at step i, and $m_i \in \{0, 1\}$ an indicator variable: equals 1 if the action step is a short trajectory and 0 if it is a long trajectory. We apply decoupled entropy constraints based on policy model's response trajectory type: (1) β_s: fixed coefficient for short paths (to prevent excessive exploration), (2) β_l: adaptive coefficient for long paths (learned dynamically).

The sample-level policy loss is defined as:

$$\beta_i = \beta_s \cdot m_i \cdot \mathbb{I}\{H_i \le H_s\} + \beta_l \cdot (1 - m_i) \cdot \mathbb{I}\{H_i \le H_l\}, \tag{2}$$

$$\mathcal{L}_{\text{p}} = \frac{1}{N}\sum_{i=1}^{N}\Big[-\min\Big(\rho_i\hat{A}_i,\ \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)\cdot\hat{A}_i\Big) - \beta_i H_i\Big], \tag{3}$$

where β_i adapts the entropy penalty per sample, $\rho_i = \pi_\theta(a_i|s_i)/\pi_{\theta_{\text{old}}}(a_i|s_i)$, H_l and H_s denote target entropy of long reasoning and short reasoning, and $\hat{A}_i$ is the estimated advantage based on reward scores in Section 3.3. The key design is the *decoupling* of entropy weights via m_i, enabling distinct regularization strategies.

Adaptive Entropy Coefficient Loss. Entropy regularization is highly sensitive to the choice of coefficient, making it difficult to select an optimal coefficient in advance. This motivates a dynamic adjustment of the entropy loss coefficient. To automatically adjust the entropy strength for long trajectories, we introduce an adaptive loss that updates β_l based on the deviation of actual entropy from a target level. The loss is computed only on steps belonging to long trajectories ($m_i = 0$) and is defined as:

$$\mathcal{L}_\beta^l = \frac{1}{\sum_j(1-m_j)}\sum_{i=1}^{N}(1-m_i)\cdot\beta_l\cdot(H_i - H_l), \tag{4}$$

where H_l is a predefined target entropy. The coefficient β_l is updated by minimizing $\mathcal{L}_\beta^L$: if $H_i < H_l$, β_l increases to encourage exploration; if $H_i > H_l$, it decreases to suppress excessive randomness. In contrast, β_s remains fixed during training.

3.3 Auto Thinking Reward Module

In this module, the model's output is evaluated using a rule-based reward (DeepSeek-AI, 2025a; Meng et al., 2025) to compute the estimated advantage for the objective loss $\mathcal{L}_{\text{p}}$. Specifically, for each question q, model generates G completions $\{o_1, o_2, \ldots, o_G\}$ using $\pi_{\theta_{\text{old}}}$. This reward module combines format and answer rewards to score each completion.

Format Reward. The format reward $\mathcal{R}_{\text{format}}(o_i) \in \{0, 1\}$ evaluates whether the output adheres to the required structural template. We define two valid reasoning modes: *think* and *no-think*, each with strict syntactic constraints:

```
[mode]think[/mode][think]reasoning process here[/think]answer
[mode]no_think[/mode][no_think]\n[/no_think]answer
```

This design encourages explicit reasoning for complex problems via the *think* mode, while allowing direct generation for simple queries via *no-think*, reducing computational overhead. During the inference stage, controllable reasoning modes are achieved by prepending special tokens to the input, as depicted at the bottom of Figure 4.

Answer Reward. We check the correctness of the tool call by comparing it against the ground-truth annotation y^*. Tool-call outputs are parsed into structured dictionaries, enabling exact matching of both the function name and all required arguments. To encourage a balance between reasoning efficiency and accuracy, we design an asymmetric reward based on the mode (*think* or *no-think*):

$$\mathcal{R}_{\text{answer}}(o_i) = \begin{cases} +1.0, & \text{if } o_i = y^*, \textit{no-think}, \\ +0.5, & \text{if } o_i = y^*, \textit{think}, \\ -0.5, & \text{if } o_i \neq y^*, \textit{think}, \\ -1.0, & \text{if } o_i \neq y^*, \textit{no-think}, \end{cases} \tag{5}$$

It incentivizes short responses when they are correct, while encouraging long reasoning when mistakes occur, prompting more careful processing in uncertain scenarios.

4 Experiments

4.1 Experimental Setup

We use the open-source Qwen2.5-7B-Instruct as our base model. We compared four baseline types: **Base**, **SFT**-trained , API-based **Frontier**, and **RLVR**-trained models. Additionally, we compare the series of models trained on the base model using *PubTool* data. See Appendix D for more details.

Evaluation Dataset. The following benchmarks are used for evaluation: (1) **BFCL** (Yan et al., 2024) provides a comprehensive dataset comprising 4k+ instances (updating), consisting of *Non-live* (with expert-curated simple tools), *Live* (with user-contributed complex tools), *Multi-turn* (with multi-turn & multi-step tool use) samples. (2) **API-Bank** (Li et al., 2023), which consists of 314 tool-use dialogues and 753 API calls. This dataset evaluates a models' abilities to correctly invoke a known API (L-1) based on a query and to retrieve and call APIs from a tool list (L-2). (3) **ACEBench** (Chen et al., 2025) is a 2k-entry benchmark for assessing agentic tool use, using its summary score in "normal" evaluation type (covering single-turn and multi-turn scenarios).

4.2 Overall Performance

The overall performance of models are shown in Table 3 and Figure 5. Firstly, the results indicate that our model consistently achieves corresponding best performance at comparable scales (~7B). For instance, compared to PubTool-SFT, AutoTool-7B with automatic think achieving +11.95 point improvement. And relative to Base model, it also has a remarkable boost with +16.43%. Secondly, our model demonstrated its more superiority in challenging scenarios (e.g., achieves +28.5% improvement compare to PubTool-SFT in *Multi-turn*). This demonstrates that our method realizes a strong robustness enhancement in complex scenarios.

Moreover, our model outperforms most SFT-trained and RLVR-trained models in BFCL, and demonstrates comparable performance with the frontier models. It also shows consistent advantageous performance on API-Bank and ACEBench compared with baselines in Figure 5. For example, on ACEBench, our model achieves a 6.5 improvement compared to GRPO and a 5.9 improvement compared to Distilled. Finally, in the inference controllable mode, when forced to think, the overall performance is on par with auto think; when forced not to think, the effect on *Multi-turn* is significantly improved compared to no-think models (e.g., PubTool-SFT).

Table 3: Comparison on the BFCL benchmark. *Overall Acc* denotes the average performance on three subsets. * indicates a single-turn tool use model; † denotes models trained on *PubTool* data with a specific method. The subscript denotes the thinking rate.

Type	Model	Non-Live	Live	Multi-Turn	**Overall Acc**
♣Base	LLaMA-3.1-8B-Instruct	84.21	61.08	9.62	50.87
	Qwen2.5-7B-Instruct	86.46	67.44	7.62	53.69
	Qwen2.5-32B-Instruct	85.81	74.23	17.75	59.67
♥Frontier	GPT-4o-2024-11-20	87.67	79.88	43.00	70.42
	o3-2025-04-16	81.42	73.43	56.12	70.32
	Gemini-2.5-Pro	89.54	76.83	30.62	65.48
♦SFT	Hammer2.1-7b(Lin et al., 2024)	88.65	75.11	23.50	61.83
	ToolACE-8B (Liu et al., 2024)	87.54	78.59	7.75	58.42
	xLAM-7b-r(Zhang et al., 2024a)	81.06	75.22	10.00	54.75
	PubTool-SFT†	88.98	77.28	9.68	58.17
	PubTool-Distilled†	87.73	78.64	15.65	60.30
♠RLVR	DeepSeek-R1-0528	75.20	77.30	38.88	63.79
	Qwen3-8B(Team, 2025a)	88.81	78.54	33	66.34
	QwQ-32B(Team, 2025b)	87.33	75.61	14.50	58.30
	Tool-N1-7B*(Zhang et al., 2025b)	89.25	80.38	-	-
	ToolRL-7B(Qian et al., 2025)	82.21	74.90	18.12	58.38
	PubTool-GRPO†	88.87	78.93	10.77	60.13
♠Ours	AutoTool-7B†	$89.76_{0\%}$	$80.22_{4.8\%}$	$38.18_{45\%}$	$70.12_{9.7\%}$
	+ *think*	89.86	80.43	39.28	70.71
	+ *no-think*	87.36	78.60	27.63	63.34

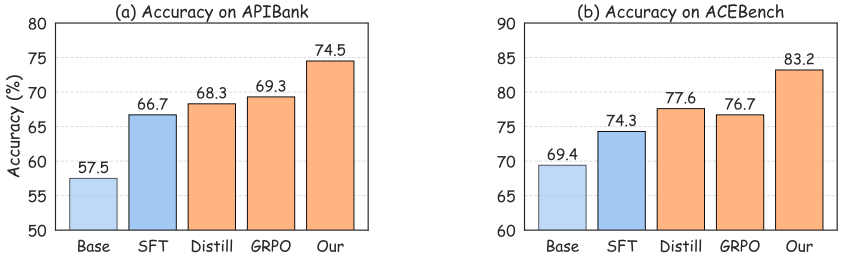


Figure 5: Performance of methods using training data *PubTool* on APIBank and ACEBench.

4.3 Deep Analysis Study

4.3.1 Ablation Study

To evaluate the effectiveness of key components in our method, we conducted an ablation study with the following variations: (1) Replaced the adaptive entropy coefficient with a fixed one (*w/o adapt coeff*); (2) Replaced the decoupling loss with a unified loss with fixed entropy constraint (*w/o decouple*); (3) Removed data quality refinement (*w/o data refine*). We also included Qwen2.5-7B-Instruct as a Base Model for comparison. As shown in Table 4, compared with the baseline, our full model delivers a significant improvement of 16.43 points in Overall performance. All components are essential to our method, and removing any component causes clear performance drops: (1) *w/o data refine* brings the largest 6.43% Overall reduction, highlighting high-quality data as a core foundation. (2) *w/o adapt coeff* leads to a 10.53% Multi-turn decline, proving its value in stabilizing multi-round interactions; (3) *w/o decouple* results in a 2.34% Overall drop, showing decoupling avoids objective interference.

Table 4: The strategy ablation performance (↑ = increase, ↓ = decrease, values are relative percentage changes from the *Our (w/. all)* model).

Models	Non-live	Live	Multi-turn	Overall
Base Model	86.46	67.44	7.62	53.69
Our (w/. all)	89.76	80.22	38.18	70.12
w/o. *data refine*	88.22 ↓1.54	73.29 ↓6.93	26.84 ↓11.34	63.69 ↓6.43
w/o. *decouple*	87.35 ↓2.41	75.98 ↓4.24	27.65 ↓10.53	64.23 ↓5.89
w/o. *adapt coeff*	88.73 ↓1.03	78.73 ↓1.49	32.14 ↓6.04	67.78 ↓2.34

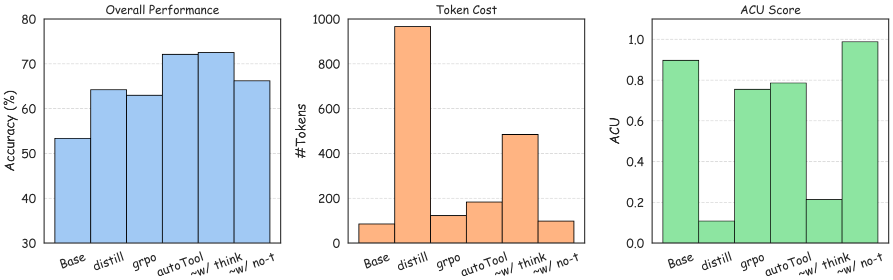


Figure 6: Inference efficiency analysis results, including performance, token cost, ACU.

4.3.2 Inference Efficiency analysis

Given the trade-off between reasoning path length, model size (~B), and performance, we introduce a new metric, Accuracy per Computation Unit (ACU), to better capture this balance and assess model inference efficiency (Ma et al., 2025). It is defined as:

$$\text{ACU} = \frac{\text{Accuracy}}{\#\text{Params} \times \#\text{Tokens}} \tag{6}$$

Since the ACU value typically falls within the range of 10^{-5} to 10^{-3}, we report it in units of 10^3 for improved readability. In addition, we report the thinking rates of our model across all submetrics.

The experimental results are summarized in Figure 6. From the results, we observe that AutoTool achieves the second-best overall performance: it reduces token cost significantly by 81%, requiring only about ~183 tokens compared to the distilled model (~966 tokens). Notably, with the forced no-think inference mode, AutoTool attains the optimal ACU score (0.97), demonstrating excellent inference efficiency. Even with the think inference mode, it still delivers the highest accuracy while cutting the token cost by half relative to the distilled model. Additionally, Table 3 shows that the our model's thinking rate reaches 45% in the Multi-Turn scenario but 0% in the No-Live scenario. The training process visualized in Appendix Section C shows our model extends reasoning trajectories for complex questions by ×5, while enabling concise responses for simple ones. This suggests the model has learned to automatically adjust the test-time scale based on sample difficulty, which effectively supports the improvement of inference efficiency.

5 Conclusion

This study focused on addressing challenges in integrating agentic LLMs with tools by optimizing the RLVR paradigm. Our research first identified two critical issues: excessive resource consumption caused by unnecessary long-trajectory reasoning, and the *reasoning collapse* phenomenon under the direct RL training, hindering effective scaling. To solve these, we proposed a decoupled adaptive entropy constraint strategy, which enables the model to automatically adjust reasoning scales based on problem difficulty, thereby balancing performance and inference efficiency. Experiments on three benchmarks confirmed the strategy's effectiveness, boosting accuracy while cutting inference token cost significantly. This work advances RL-based agentic tool-use training and provides a practical auto-scaling solution for efficiently handling tasks.

Ethics Statement

This work strictly adheres to the ICLR Code of Ethics: it involves no human subjects, uses datasets compliant with original licensing agreements (ensuring privacy and legal compliance), and avoids discriminatory biases in experimental design/results; all authors confirm adherence, with no conflicting sponsorships.

Reproducibility Statement

For reproducibility, key details are referenced across the main text (methodology Section 3.2, experimental setup Section 4.1), appendix (hyperparameters Section D, data processing details Section B, full prompt Section F), and supplementary materials (anonymous source code). We ensure data splits, random seeds, and environment configurations are explicitly stated, allowing researchers to independently verify our findings under identical conditions.

References

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, et al. Acebench: Who wins the match point in tool learning? *arXiv e-prints*, pp. arXiv–2501, 2025.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025a. URL `https://arxiv.org/abs/2501.12948`.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025b. URL `https://arxiv.org/abs/2501.12948`.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: Llm learns when to think. *arXiv preprint arXiv:2505.13379*, 2025.

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. *arXiv preprint arXiv:2504.11536*, 2025.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025.

Shijue Huang, Hongru Wang, Wanjun Zhong, Zhaochen Su, Jiazhan Feng, Bowen Cao, and Yi R Fung. Adactrl: Towards adaptive and controllable reasoning via difficulty-aware budgeting. *arXiv preprint arXiv:2505.18822*, 2025.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

Teknium interstellarninja. Hermes-function-calling-dataset-v1, 2024. URL `https://huggingface.co/NousResearch/hermes-function-calling-v1`.

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.

Kimi K2. Kimi-researcher: End-to-end rl training for emerging agentic capabilities. https://moonshotai.github.io/Kimi-Researcher/, 2024. Accessed: 2025-09-10.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*, 2022.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, 2023.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*, 2025a.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025b.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, et al. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*, 2024.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, et al. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*, 2024.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*, 2025.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, et al. Mm-eureka: Exploring visual aha moment with rule-based large-scale reinforcement learning. *CoRR*, 2025.

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

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

OpenAI. o3. https://openai.com/index/introducing-deep-research/, 2025. Accessed: Sep 10, 2025.

Jie Ouyang, Ruiran Yan, Yucong Luo, Mingyue Cheng, Qi Liu, Zirui Liu, Shuo Yu, and Daoyu Wang. Training powerful llm agents with end-to-end reinforcement learning, 2025. URL https://github.com/0russwest0/Agent-R1.

Zhenyu Pan and Han Liu. Metaspatial: Reinforcing 3d spatial reasoning in vlms for the metaverse. *arXiv preprint arXiv:2503.18470*, 2025.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, et al. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601*, 2025.

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. *arXiv preprint arXiv:2504.13958*, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*, 2023.

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):198343, 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, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, et al. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*, 2022.

Linxin Song, Jiale Liu, Jieyu Zhang, Shaokun Zhang, Ao Luo, Shijian Wang, Qingyun Wu, and Chi Wang. Adaptive in-conversation team building for language model agents. *arXiv preprint arXiv:2405.19425*, 2024.

Qwen Team. Qwen3 technical report, 2025a. URL `https://arxiv.org/abs/2505.09388`.

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

Jiaqi Wang, Kevin Qinghong Lin, James Cheng, and Mike Zheng Shou. Think or not? selective reasoning via reinforcement learning for vision-language models. *arXiv preprint arXiv:2505.16854*, 2025.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.

Shijie Xia, Yiwei Qin, Xuefeng Li, Yan Ma, Run-Ze Fan, Steffi Chern, Haoyang Zou, Fan Zhou, Xiangkun Hu, Jiahe Jin, et al. Generative ai act ii: Test time scaling drives cognition engineering. *arXiv preprint arXiv:2504.13828*, 2025.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Berkeley function calling leaderboard. 2024.

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

Yuanqing Yu, Zhefan Wang, Weizhi Ma, Zhicheng Guo, Jingtao Zhan, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. Steptool: A step-grained reinforcement learning framework for tool learning in llms. *arXiv preprint arXiv:2410.07745*, 2024.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025a.

Weihao Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. `https://hkust-nlp.notion.site/simplerl-reason`, 2025b. Notion Blog.

Yirong Zeng, Xiao Ding, Yuxian Wang, Weiwen Liu, Wu Ning, Yutai Hou, Xu Huang, Bing Qin, and Ting Liu. Boosting tool use of large language models via iterative reinforced fine-tuning. *arXiv preprint arXiv:2501.09766*, 2025c.

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

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Awalgaonkar, Rithesh Murthy, Eric Hu, Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*, 2024a.

Shaokun Zhang, Jieyu Zhang, Dujian Ding, Mirian Hipolito Garcia, Ankur Mallick, Daniel Madrigal, Menglin Xia, Victor Rühle, Qingyun Wu, and Chi Wang. Ecoact: Economic agent determines when to register what action. *arXiv preprint arXiv:2411.01643*, 2024b.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*, 2025b.

Wang Zihan, Wang Kangrui, Wang Qineng, Zhang Pingyue, Li Linjie, Yang Zhengyuan, Jin Xing, Yu Kefan, Minh Nhat Nguyen, Liu Licheng, Eli Gottlieb, Yiping Lu, Cho Kyunghyun, Wu Jiajun, Fei-Fei Li, Wang Lijuan, Choi Yejin, and Li Manling. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Use of LLM

LLMs (GPT-4o) were only used as general-purpose tools to draft baseline literature summaries and proofread minor grammar, no LLM contributed to core ideation, algorithm development, analysis, or writing, and all LLM-assisted content was verified for accuracy/integrity. No LLM is eligible for authorship.

A Related Work

A.1 Agentic Tool-Use

Enhancing LLMs with external tools has emerged as a pivotal direction for addressing complex tasks in open domains (Qu et al., 2025; Wang et al., 2024). Typical applications include integrating LLMs with search engines (Zhang et al., 2024b; Lazaridou et al., 2022; Shuster et al., 2022), calculators (Nakano et al., 2021), and Python interpreters (Wang et al., 2024; Song et al., 2024; Chen et al., 2022). Three common paradigms are widely adopted for training tool-use LLMs: (1) SFT: imitates the reasoning patterns from labeled high-quality examples, enabling models to learn standard tool-use workflows (Liu et al., 2024; Zhang et al., 2024a; Qin et al., 2023; Prabhakar et al., 2025). (2) RL with direct preference optimization: aligns model tool-use behavior with human intentions by optimizing against human preference signals (Zeng et al., 2025c; Yu et al., 2024). (3) RL with Verifiable Rewards (RLVR): as a novel approach, leverages scalable test-time inference and utilizes verifiable signals as rewards to refine the model's tool-use decisions (Li et al., 2025b).

A.2 RL Scale-up

Reinforcement learning (RL) has gained traction as a more scalable and generalizable training paradigm. Models like R1-Zero leverage group relative policy optimization (GRPO) (Shao et al., 2024) to unlock the model's reasoning capabilities at test time (DeepSeek-AI, 2025a; Yu et al., 2025). This R1-style reasoning paradigm, marking a shift from train-time scaling to test-time scaling (Muennighoff et al., 2025; Xia et al., 2025), has demonstrated success in mathematics (Shao et al., 2024), coding (Pan & Liu, 2025), and agentic tool use (Feng et al., 2025; Jin et al., 2025).

Recently, several works have explored automatic scaling , i.e., enabling models to adaptively select the optimal reasoning mode based on problem difficulty (Fang et al., 2025; Zhang et al., 2025a; Huang et al., 2025; Wang et al., 2025). In agentic tool-use tasks, auto-scaling is particularly critical: many such problems can be solved with short reasoning, whereas excessively long reasoning leads to unnecessary resource consumption. While RL-based scaling for tool use in open-domain reasoning has been investigated (Zhang et al., 2025b; Qian et al., 2025), RL with auto-scaling remains unexplored in agentic tool use.

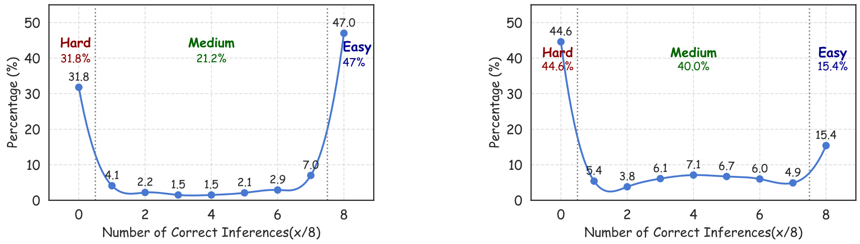
B Details in Data Preparation

Source of Training Data Details. The raw data was sourced as follows:

- ToolACE (Liu et al., 2024): A general tool-use dataset teaching models when to invoke tools vs. respond directly, enhancing multi-step decision-making.
- xLAM (Zhang et al., 2024a; Prabhakar et al., 2025): A compositional dataset requiring one or more tool calls per turn. We mixed the original 60k xLAM with its multi-turn variant APIGen-MT-5k (Prabhakar et al., 2025).
- Hermes Function-Calling (interstellarninja, 2024): Designed to train LLMs in function calls and structured output from natural language. We extracted function call-related samples.

The dataset features various conversational scenarios where AI agents are required to interpret queries and execute appropriate single or multiple function calls. In Section 2, data distillation employs Deepseek-R1-0528 (DeepSeek-AI, 2025a). Subsequently, in Section §3.1, to mitigate model bias by aligning with a no-think model, data distillation is carried out using Qwen3-32B (Team, 2025a).

Data Processing Pipeline & Distribution Details. We obtained PubTool from raw data through following data processing workflow: (1) We randomly downsampled xLAM to balance the sample sizes across the three datasets. (2) We removed overly simple and excessively difficult samples; Figure 7 shows the raw-data distribution of successful reasoning counts (pass@8). Guided by this distribution, we partitioned the data into hard (31.8%), medium (21.2%), and easy (47%) subsets (Figure 7a) and re-balanced the difficulty distribution accordingly. (3) For RL data, we further refined the set by prioritizing samples that align closely with the model's current learning trajectory (details in the next paragraph). For comparison, we visualized the PubTool distribution in the same way (Figure 7b). We observed that the original corpus is concentrated in the easy and hard extremes, whereas PubTool peaks in the hard subset and is sparse in the easy subset. We argue that training on data of moderately high difficulty better elicits the model's test-time scaling capability (He et al., 2025).



(a) Data Distribution of Raw data

(b) Data Distribution of PubTool

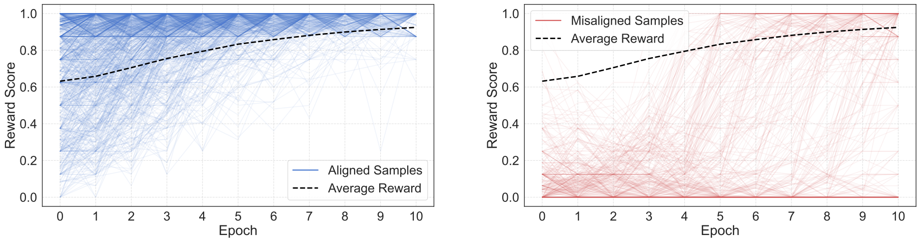
Figure 7: The number of correct inferences distribution with performing 8 rounds of reasoning on the raw training data (a). The distribution of PubTool after data processing (b).

RL Data Refine Details. In the second phase of data processing, we prioritize training samples by their alignment with model learning trajectories, measured through the variance of reward scores from the mean, lower variance indicates better alignment. Better alignment corresponds to lower variance of reward scores, defined as:

$$\text{Var}(r) = \frac{1}{n-1}\sum_{i=1}^{n}(r_i - \mu_r)^2, \quad \mu_r = \frac{1}{n}\sum_{i=1}^{n} r_i$$

where lower Var(r) indicates better alignment. This sampling result is illustrated in Figure 8. From the figure, we observe that the average reward ranges between 0.7 and 0.9, with aligned samples showing higher scores in the upper-left region and misaligned samples displaying lower scores.

Effectiveness of Data Refinement. We experimentally verified its effectiveness. After warm-up SFT, Figure 9 shows GRPO training processes with and without data refinement. Results indicate



(a) Retained: with aligned samples

(b) Removed: with misaligned samples

Figure 8: We retain aligned samples (i.e., those with low variance (a)) and remove misaligned samples (i.e., those with high variance (b)).

data refinement increases accuracy reward score by +15%, reduces training fluctuation variance, and enhances stability. Additionally, the model's thinking rate converged to a lower level, indicating improved memory capacity. BFCL evaluation results show GRPO with data refinement reached 66.82%, versus 60.78% without, an improvement of +6.04%. These enhancements are attributed to data refinement filtering substantial noise while retaining high-contribution samples.

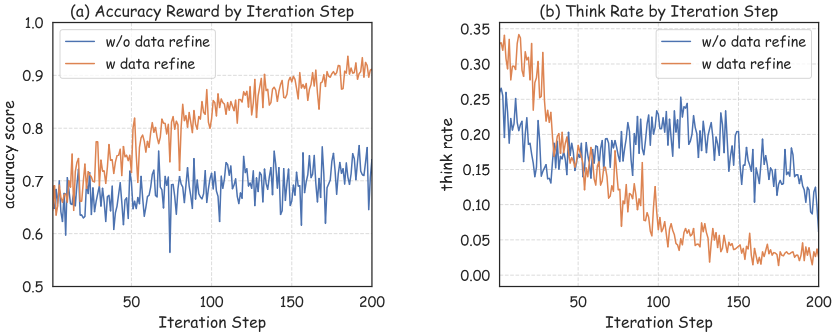


Figure 9: RL training processes (with and without data refinement) are shown, along with accuracy scores and thinking rates.

C Visualization of Training Dynamics

To demonstrate auto-scaling effects, we visualized the training process (Figure 10). As training progressed, accuracy improved while the thinking rate gradually decreased to 5%, indicating fewer problems required long reasoning, suggesting enhanced intrinsic tool-using capabilities. Additionally, response length and entropy achieved decoupled control: think mode enabled 500% longer reasoning trajectories than no-think mode, with corresponding higher actor entropy reflecting greater exploration tendency. These visualizations confirm that training enhanced tool-using abilities and successfully enabled auto test-time scaling based on problem difficulty and model proficiency.

D Complementary Experiments

D.1 More Implementation Details

The experiments were executed using the publicly accessible training framework MindSpeed-RL[2], an end-to-end reinforcement learning acceleration framework based on the Ascend ecosystem. The BFCL is an evolving benchmark and we utilized the version checked out on June 14, 2025. For the

[2] https://gitee.com/ascend/MindSpeed-RL

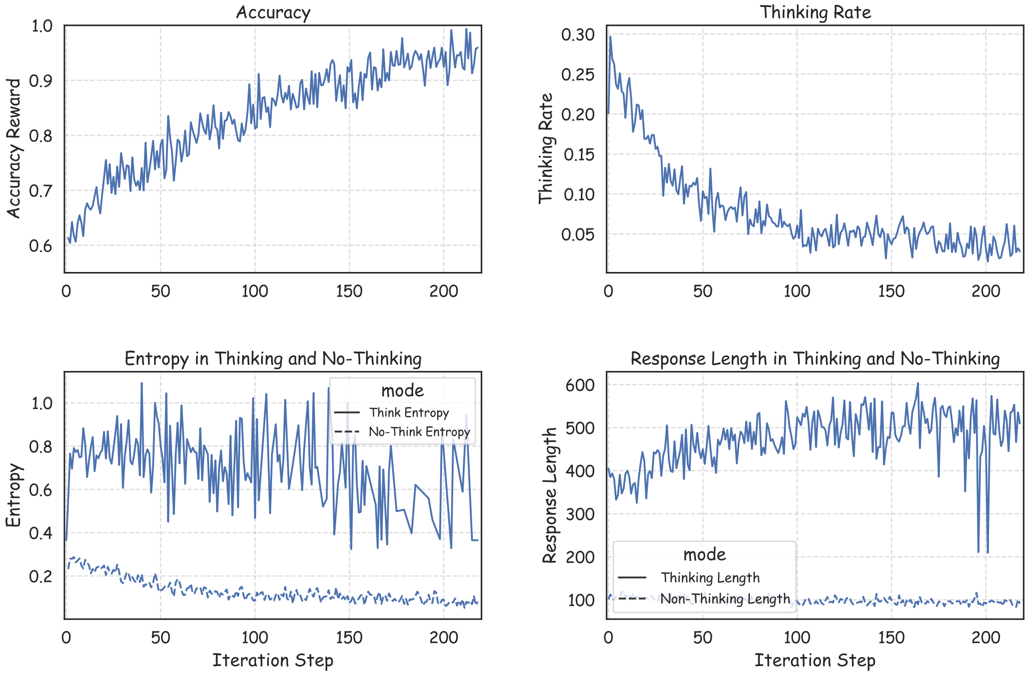


Figure 10: The visualization of training dynamics.

training model, we selected the best performance checkpoint on the valid dataset. In the Test-time Scale paradigms analysis (§2), we used an instruct model for SFT and a base model for GRPO. We employ a full-parameter training strategy for all SFT. In baseline trained on PubTool data, we trained on the complete dataset using specific methods (e.g., SFT, Distilled SFT, and GRPO). In the PubTool-GRPO training, we adopted the widely used think prompt pattern, which follows the format: `<think> reasoning process here </think><answer> answer here </answer>`. Each RL training run for the 7B model completed within 4 hours on a cluster of 32 Ascend 910b NPUs (configured as 4 nodes × 8 NPUs). The hyperparameters used are detailed in Table 5.

Baselines (1) *Base Model*: the original model without additional training (e.g., Qwen2.5-series, LLaMA3.1-series). (2) *SFT-trained Model*: ToolACE-8B (trained on the full ToolACE dataset (Liu et al., 2024)), xLAM-series (trained on the full xLAM dataset (Zhang et al., 2024a)), and Hammer-series (trained on xLAM with function masking (Lin et al., 2024)). (3) *API-based* closed-source frontier models (e.g., GPT-series, Gemini-series). (4) *RLVR-trained Model*: models trained using GRPO as the RL paradigm, such as QwQ-32B (Team, 2025b), Qwen3-series (Team, 2025a), Tool-N1 series (single-turn tool-use models trained on mixed ToolACE and xLAM data (Zhang et al., 2025b)), and ToolRL(trained in subset of mixed ToolACE and xLAM data (Qian et al., 2025)).

E Hyperparameter Analysis

Model performance appears sensitive to target entropies and the initial choice of penalty coefficient β. To identify a suitable target entropy for entropy constraints, we conducted a hyperparameter analysis.

F Prompt Design for Auto Think

To explore a suitable prompt design for Auto Think, we conducted a preliminary analysis of the four kind of prompts listed below:

Hyperparameter	Value	Hyperparameter	Value
Data Configuration		**RL Optimization**	
Global Batch Size	128	Learning Rate	1e-6
Max Prompt Length	12000	LR Decay Style	constant
Max Response Length	2048	Mini Batch Size	128
Micro Batch Size	4	KL Loss Used	False
Train Steps	200		
Rollout Configuration		**Entropy Constraints**	
Rollout Name	vllm	Clip Higher ϵ	0.28
GPU Memory Utilization	0.5	Think Target Entropy H_l	0.2
Number of Rollouts	8	No-Think Target Entropy H_s	0.1
Temperature	1.0	Init Adaptive Coefficient β_l	0.1
Tensor Model Parallel Size	1	Fixed Coefficient β_s	0.1
Top_P	1.0		

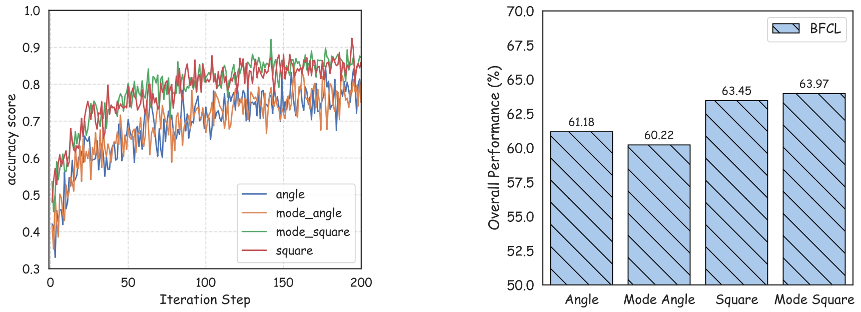
Table 5: The configurations for RL training with GRPO.

- Controlled reasoning mode with square tags: **[mode]no_think[/mode] [no_think] \n [/no_think] [tool_call] tool calls here [/tool_call]**
- Uncontrolled reasoning mode with square tags: **[no_think]\n[/no_think] [tool_call] tool calls here [/tool_call]**
- Controlled reasoning mode with angle tags: **<mode>no_think</mode> <no_think> \n </no_think> <tool_call> tool calls here </tool_call>**
- Uncontrolled reasoning mode with angle tags: **<no_think>\n</no_think> <tool_call> tool calls here </tool_call>**

We trained the model starting from Qwen2.5-7B-Instruct using the original GRPO algorithm with PubTool LRL data. Their training processes and evaluation results are presented in Figure 11. From the results, two key observations emerge: (1) Square tags ([ ]) exhibit better adaptability than angle tags (<>). This may be because the model used angle tags for segmentation in the pre-training phase, reusing these tags for a different purpose (reasoning mode control) is likely to cause signal interference. (2) Additionally, the explicit "reasoning mode" prefix does not obviously affect performance. The evaluation results show that the controlled reasoning mode with square tags achieves the best performance; thus, we adopt this prompt design for auto scaling.

G LIMITATION AND FUTURE WORK

Despite promising results in tool-use scenarios, our method has latent concerns to clear. First, we only tested it on a specific model size, future work should verify its scalability across different model scales (e.g., 3B, 13B, 32B parameters) and architecture series, e.g., LLaMA-series. Second, our method's generalizability beyond tool-use tasks is unproven. It is valuable to evaluate its performance on other complex reasoning tasks (e.g., mathematics, logical deduction) to confirm if it can similarly enhance reasoning steps or reduce computational costs. Third, our method currently relies on a specific RL algorithm. Future research should test its compatibility with other RL algorithms (e.g., PPO (Schulman et al., 2017; Engstrom et al., 2020) and DAPO (Yu et al., 2025)) to verify if the decoupled entropy constraint strategy is effective across different algorithmic paradigms. We will address these limitations in future work.



(a) Accuracy Reward by Iteration Step

(b) Performance Comparison of Four Prompt Design

Figure 11: Visualization of training processes and evaluation results for four prompt designs.

The Full System Prompt for Automatic Think in RL

You are an advanced function composition agent. Your goal is to solve user queries efficiently. In the multi-turn dialogue loop: the interaction is a cycle: You [tool_call], you receive a [tool_response], and you MUST use that new information to plan your next step.

# Tools
You are provided with function signatures within [tools] and [/tools] tags:
[tools] {functions} [/tools]

# Action Phase
1. Choose an Action Mode: For every turn, you MUST start your response by choosing an action mode (think vs no_think) based on the task's complexity.
no - think: Use for complex reasoning. Enclose your detailed thought process within [think] and [/think] tags.
no - no_think: Use for simple, straightforward tasks. You MUST use an empty, self-closing block: [no_think]\n[/no_think].
Your response MUST begin by enclosing the selected mode name within [mode] and [/mode] tags.

2. Decide on an Action Path: After that, you must choose ONE of the following two paths:

## Path A: Call Functions
WHEN: The user's intent is tool-related and you have all required functions and parameters.
The tool_calls field is a JSON object with function names and arguments within [tool_call] and [/tool_call] XML tags. i.e., [tool_call] [{"name": <function-name>, "arguments": <args-json-object>}, {"name": <function-name2>, "arguments": <args-json-object2>}, ...] [/tool_call]

EXAMPLE:
[mode]no_think[/mode] [no_think]\n[/no_think] [tool_call] tool calls here [/tool_call]
EXAMPLE:
[mode]think[/mode] [think] reasoning process here [/think] [tool_call] tool calls here [/tool_call]

## Path B: Respond Directly to the User
WHEN: You need to provide a natural language text response. This happens in three main scenarios:
(1) After receiving tool execution feedback enclosed within [tool_response] and [/tool_response] tags, continue to respond to user queries based on this feedback.
(2) The user's query is general conversation and not related to any tool.
(3) Ask for more information if the given conversational context lacks the required functions or parameters.

EXAMPLE:
[mode]no_think[/mode] [no_think]\n[/no_think] natural language sentences you talk with user
EXAMPLE:
[mode]think[/mode] [think] reasoning process here [/think] natural language sentences you talk with user

Figure 12: System Prompt Design for Automatic Scaling Tool-Use in Multi-Turn Dialogue.