ToolSample: Dual Dynamic Sampling Methods with Curriculum Learning for RL-based Tool Learning

Anonymous ACL submission

Abstract

Reinforcement learning (RL) is increasingly used for LLM-based tool learning. However, its efficiency is often hampered by an overabundance of samples that contribute little to the model's improvement as training progresses. Existing RL dynamic sampling methods are typically based on binary (success/fail) rewards. Such methods are ill-suited for tool learning task characterized by interdependent sub-tasks of varying difficulties, which can yield multiple fine-grained rewards. To address this gap, we introduce Dynamic Sampling with Curriculum Learning (DSCL), which leverages the complex sub-tasks and reward dynamics inherent to tool learning. DSCL features two core components: Reward-Based Dynamic Sampling, which uses multi-dimensional reward statistics to prioritize valuable data, and Task-Based Dynamic Curriculum Learning, which adaptively focuses training on less-mastered sub-tasks. Through extensive experiments on the widely used tool learning benchmarks, we demonstrate that this tailored approach significantly improves model performance, outperforming strong baselines with a 4.16% gain on the BFCL V3 dataset and 3.85% gain on the API-Bank dataset. Our method will be publicly available at http://anonymous.com/.

1 Introduction

Reinforcement Learning (RL) has emerged as a key strategy to enhance Large Language Models (LLMs) for tool learning (Qin et al., 2024; Kumar et al., 2025; Qu et al., 2025; Guo et al., 2025), which aims to use external APIs to complete complex human instructions (Tang et al., 2023; Zeng et al., 2025; Feng et al., 2025a; Chen et al., 2025; Li et al., 2025). To better leverage RL in tool learning, many efforts have been devoted to tailoring RL and improving performance by designing fine-grained rewards for interdependent sub-tasks such as tool selection, tool parameter key extraction, and parameter value completion (Qian et al., 2025; Zhang et al., 2025; Feng et al., 2025a).

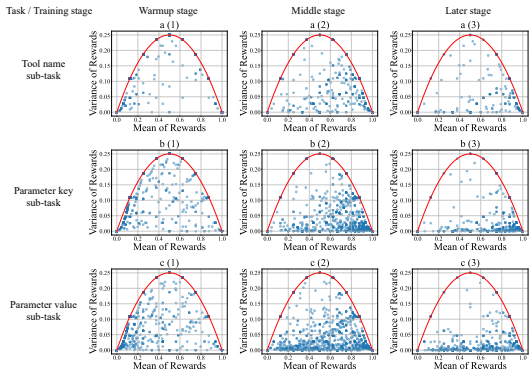

Figure 1: Mean-Variance Scatter Plots of training rewards across different stages of RL (GRPO) training. (Warmup: The stage where the model's format prediction is almost correct; Middle & Later: The middle and final stages of training, respectively.) Key observations include: 1) In the Middle and Later stages, the reward variance for many samples collapses towards zero; 2) Unlike traditional binary (success/fail) rewards, where variance is a direct function of the mean (red line), ToolRL's mean-variance distribution exhibits a wide range of variances for any given mean (blue plots); 3) In the same stage, different sub-tasks exhibit asynchronous convergence (e.g., a(2) and c(2)).

Although effective, these approaches still exhibit a diminishing focus on under-learned samples as the model accuracy improves throughout RL training. As shown in Figure 1, during the middle to later stages of RL's training, a significant portion of samples exhibit near-zero reward variance, particularly for parameter-related predictions (subfigure b(2-3) and c(2-3)). Such homogeneous samples provide limited learning benefit, and an excessive proportion can even impede further model improvement (Team et al., 2025; Bae et al., 2025). This highlights a key bottleneck in current RL-based tool learning: wasting computational resources on "easy" samples that no longer contribute effectively

to training.

To address this inefficiency, dynamic sample selection (i.e., DAPO (Yu et al., 2025)) has proven effective in other RL tasks like mathematics (Yu et al., 2025) and code generation (Liu et al., 2025). However, these methods are ill-suitable for tool learning, as it presents two unique challenges: 1) Complex Multi-Task Structure: Tool learning is a composite of interdependent sub-tasks (e.g., tool selection, parameter completion). As illustrated in Figure 1 (e.g., subfigures a(2) and c(2)), these sub-tasks often converge asynchronously, meaning a sample that is "easy" for one sub-task might still be valuable for another. 2) Fine-Grained, Multi-Valued Rewards: Unlike tasks with binary (success/fail) rewards, tool learning benefits more from fine-grained, multi-valued reward mechanisms (Qian et al., 2025). This fundamentally alters the statistical properties of the samples. As shown in Figure 1, while a binary reward mathematically locks variance to the mean (red curves), a multi-valued reward decouples them (blue scatter plots). This decoupling implies that the reward mean and variance can serve as independent signals for evaluating sample value, a property untapped by existing methods.

Therefore, an effective dynamic sampling method for tool learning must be more sophisticated. It needs to: (1) leverage multidimensional signals (i.e., both reward mean and variance) to assess sample value; (2) dynamically adjust its focus based on the asynchronous convergence of different sub-tasks; and (3) balance the sample distribution across sub-tasks to ensure holistic model improvement.

To meet these requirements, we propose Dynamic Sampling with Curriculum Learning (DSCL), a novel methodology designed to fully exploit the potential of fine-grained rewards in tool learning. DSCL integrates two synergistic strategies: 1) Reward-Based Dynamic Sampling (RDS): Instead of relying on a single metric, RDS assesses data from three dimensions: the reward mean, the reward variance of a group of rollouts, and the evolution of these statistics over the training history. This provides a holistic assessment of sample difficulty, stability, and long-term learning trajectory. 2) Task-Based Dynamic Curriculum Learning (TDCL): TDCL adaptively prioritizes sub-tasks that require more training, ensuring that the model focuses on samples with the highest potential for improvement. Overall, this paper offers the following contributions:

- We identify the sample inefficiency problem in RL-based tool learning and introduce DSCL, a novel framework that pioneers curriculum learning for dynamic data sampling to address this.
- We propose a two-component methodology within DSCL: RDS, which dynamically samples data using multiple dimensions of the mean and variance of rewards; TDCL, which adaptively manages the asynchronous convergence of interdependent sub-tasks.
- We conduct comprehensive experiments to evaluate our DSCL method against several strong baselines. The results on the BFCL V3 and API-Bank, supported by our in-depth analysis, confirm its significant effectiveness and superiority.

2 Related Work

Our research builds upon recent advancements in two key areas: the development of tool learning and the optimization of the training process in RL.

2.1 Tool Learning

The ability of LLMs to interact with external tools has evolved significantly. Foundational work like ToolLLM (Qin et al., 2023) pioneered the field by creating large-scale benchmarks for supervised fine-tuning. To overcome the limitations of static imitation, subsequent research has employed Reinforcement Learning to optimize sequential tool use, as seen in StepTool (Yu et al., 2024), Search-R1 (Jin et al., 2025), and Tool-N1 (Zhang et al., 2025), etc. The success of RL, however, is highly dependent on effective reward engineering; previous works such as ToolRL (Qian et al., 2025) and ARTIST (Singh et al., 2025) focus on systematically addressing this challenge.

2.2 Sample Selection for RL

To improve the efficiency of RL-based training, curriculum learning and dynamic sampling strategies are two widely used strategies. A core idea of Curriculum Learning is to structure training from simple to complex tasks, as demonstrated by Confucius (Gao et al., 2024). A prominent strategy within this is difficulty-aware sampling, which dynamically focuses on problems of appropriate difficulty. Its effectiveness was shown at scale by

Kimi K1.5 (Team et al., 2025), with focused studies like RCS (Feng et al., 2025b) and online difficulty filtering (Bae et al., 2025) confirming that an intermediate difficulty level maximizes learning efficiency. Dynamic sampling organizes samples from an alternative perspective. Approaches like POLARIS (An et al., 2025) employ adaptive filtering to enhance exploration, while frameworks such as DAPO (Yu et al., 2025) and VAPO (Yue et al., 2025) integrate dynamic sampling directly into the policy optimization process to address scaling challenges. Razin et al. (2025) have also theoretically explored the detrimental effects of low-variance rewards on training effectiveness.

Unlike existing RL dynamic sampling methods, we produce a novel method including two strategies. Our method is specifically tailored for tool learning task by evaluating data from multiple dimensions and employing curriculum training based on sub-task difficulty. This synergistic approach enables dynamic sampling strategies to further achieve a better performance on this task.

3 Reward Design for Tool Learning

Qian et al. (2025) have done a comprehensive study on reward design. They have divided the tool learning task into four sub-tasks for GRPO training. We maintain their definitions of rewards and modify each sub-task into an independent representation. For a specific query, the reward functions compare the predicted tool calls $\widehat{Y} = \{\hat{y}_1, ..., \hat{y}_m\}$ with the ground-truth tools $Y = \{y_1, ..., y_m\}$. It includes the following four components:

Format Reward The format reward $R_{format} \in \{0, 1\}$ checks whether the model's response meets the requirement for both content and order:

$$R_{format} = \begin{cases} 1, & \text{if the format meet all requirements} \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

Tool Name Reward The tool name reward R_{name} evaluates the correctness of each tool's name. The $N_{\widehat{Y}}$ and N_Y represent the predicted tool names and the ground truth, respectively.

$$R_{name} = \frac{|N_{\widehat{Y}} \cap N_Y|}{|N_{\widehat{Y}} \cup N_Y|} \in [0, 1]. \tag{2}$$

Parameter Key Reward The parameter key reward $R_{key} \in [0, |Y|]$ checks whether the key of the parameter of each tool is correct, which is on the premise that the tool name is correct:

$$R_{key} = \sum_{y_i \in Y} \frac{|K(f(y_i)) \cap K(y_i)|}{|K(y_i)| + |K(f(y_i)) - K(y_i)|}, \tag{3}$$

where function $f(y_i)$ selects the best matches tool $\hat{y}_i$ with y_i in the predicted tools. If there is no match, it returns empty. The K represents the set of keys of a specific tool.

Parameter Value Reward The parameter value reward $R_{value} \in [0, \sum_{y_i \in Y} |K(y_i)|]$ evaluates the correctness of parameter values for all the matching keys.

$$R_{value} = \sum_{y_i \in Y} \sum_{k \in K(y_i)} \frac{\mathbb{1}[V(f(y_i)^k) = V(y_i^k)]}{|V(y_i)| + |V(f(y_i)) - V(y_i)|}, \tag{4}$$

where $V(y_j^k)$ represents the value that is corresponding to the tool y_j's key k.

Total Reward The final reward first maps the sum of the rewards of the last three sub-tasks to the interval $[-3, 4]$, and then adds it with R_{format} to get the final reward R.

$$R = R_{format} + \mathcal{M}_{-3}^{3}(R_{name} + R_{key} + R_{value}), \tag{5}$$

$$\mathcal{M}_{min}^{max}(R) = \left\{ \frac{max - min}{R_{max} - R_{min}} * (r_i - R_{min}) + min \right\}, \tag{6}$$

where $\mathcal{M}$ represents the mapping function, r_i represents an element in the set R, R_{max} and R_{min} represent the upper and lower bounds of the set R.

4 Methodology

In this section, we will introduce our proposed Dynamic Sampling with Curriculum Learning (DSCL) for RL-based tool learning. The DSCL method contains two strategies, Reward-Based Dynamic Sampling (RDS) and Task-Based Dynamic Curriculum Learning (TDCL). We will introduce these two methods separately and then explain how they constitute the DSCL method.

4.1 Reward-Based Dynamic Sampling (RDS)

We will first describe the basic formulation of the tool learning task. Let $D = \{d_1, ..., d_k\}$ represents the dataset for training. For each data d_i at the j-th epoch, we generate G samples and calculate their rewards $\mathcal{R}^{i,j} = \{R_1^{i,j}, ..., R_G^{i,j}\}$.

The RDS method is initiated only after the model

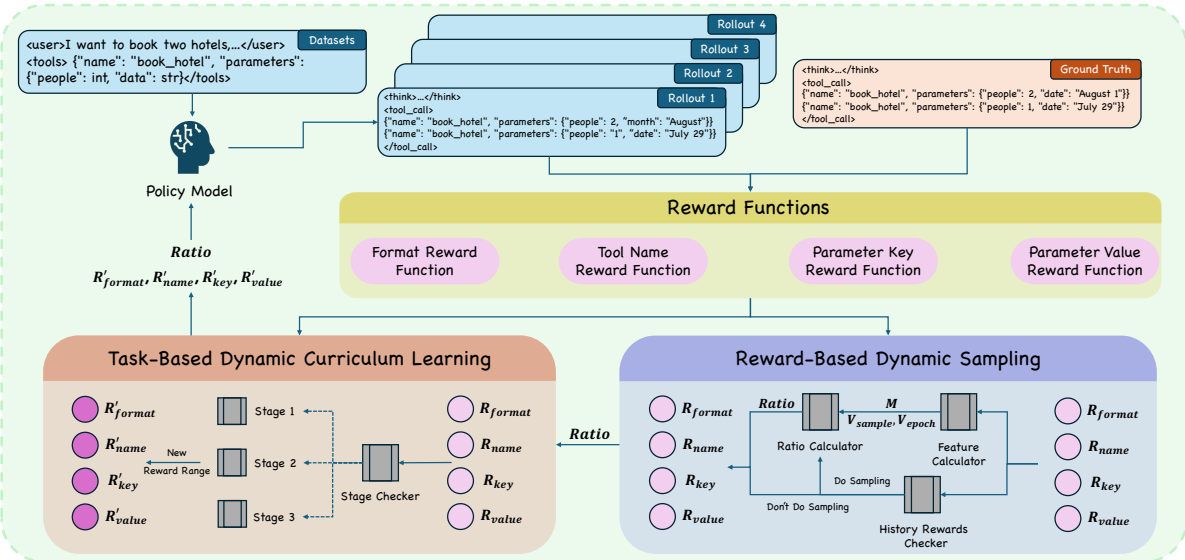

Figure 2: The training framework of our proposed method. For all rollout samples of each training data, we first calculate its reward on each sub-task. Based on the rewards, we use the Reward-Based Dynamic Sampling method to first calculate the probability of sampling this data, and then use the Task-Based Dynamic Curriculum Learning method to adjust the reward range of each sub-task according to different training stages. Finally, the processed reward is passed to the model for update.

achieves stable proficiency in tool prediction[1], a strategic delay designed to prevent the premature rejection of a large volume of samples. Subsequently, it quantifies sample importance by integrating a sample's current training status with the potential learning advantage it offers, thereby enabling dynamic sampling across the entire training corpus. To achieve this, for each data d_i at the j-th epoch, we select the following three indicators to capture the training state and reward changes, so as to better dynamically sample the data:

Mean Reward of Samples Mean reward for all rollouts:

$$M^{i,j} = Mean(\{R_1^{i,j}, ..., R_G^{i,j}\}). \tag{7}$$

Variance of Samples The sample-level variance V_{sample} for each data at each epoch:

$$V_{sample}^{i,j} = Var(\{R_1^{i,j}, ..., R_G^{i,j}\}). \tag{8}$$

Variance of Epoches The epoch-level variance, i.e., the variance of all $M^{i,j}$ during past epochs:

$$V_{epoch}^{i,j} = Var(\{M^{i,1}, ..., M^{i,j}\}). \tag{9}$$

Depends on these three indicators, we have dynamically divided the data into the following four categories:

- a. Easy data: $M^{i,j} = 4$.
- b. Hard data: $M^{i,j} < t_{mean}$.
  - b.1 $V_{sample}^{i,j} > t_{var}$ or $V_{epoch}^{i,j} > t_{var}$.
  - b.2 $V_{sample}^{i,j} < t_{var}$ and $V_{epoch}^{i,j} < t_{var}$.
- c. Intermediate data: $M^{i,j} \in (t_{mean}, 4)$.
  - c.1 $V_{sample}^{i,j} > t_{var}$ and $V_{epoch}^{i,j} > t_{var}$.
  - c.2 $V_{sample}^{i,j} < t_{var}$ or $V_{epoch}^{i,j} < t_{var}$.

We assign different importance as the sampling ratio according to the category of each data d_i:

$$Ratio_i = \begin{cases} 0.0, & d_i \in \{a, b.2\} \\ 0.5, & d_i \in \{c.2\} \\ 1.0, & d_i \in \{b.1, c.1\} \end{cases}. \tag{10}$$

Specifically, we discard samples that are either too challenging for the current model or have already been fully mastered. For challenging samples, we fully retain them if they exhibit either effective diversity across rollouts or exploratory behavior from the model compared to past epochs. For less challenging samples, however, we apply a stricter requirement, retaining them only if they satisfy both conditions. Other samples are partially retained, as relying exclusively on highly informative or exploratory samples can lead to training instability (Dang and Ngo, 2025). Furthermore, t_{mean}

[1]We operationalize this stability criterion as the mean reward of seven consecutive batches surpassing a threshold of 1.0 in practice.

and t_{var} are treated as hyperparameters that are adjusted during training. This allows the method to accommodate the sensitivity of the reinforcement learning process to variations in data and model states (Wang et al., 2025).

4.2 Task-Based Dynamic Curriculum Learning (TDCL)

The sub-tasks within the tool learning framework exhibit varying levels of difficulty and a natural progressive relationship. For instance, the format task is foundational, providing structured data for all subsequent operations. We further analyze the training loss and find a difficulty hierarchy: 1) Extracting the tool name and its parameter keys is usually synchronous, i.e., successfully predicting the tool name will link it to the corresponding parameter keys; 2) Accurately extracting parameter values is significantly more difficult.

Based on these observations, we design a three-stage dynamic curriculum learning strategy. At each stage, we adjust the reward weights for these sub-tasks, as defined in Eq. 6, to systematically shift the training focus from simpler to more complex objectives:

- **Stage 1:** We increase the λ_{format} to 2.5 times its original value and reduce other three parameters by half and the penalty to enhance the learning of the format in the first stage.

$$R = 2.5 * R_{format} + \mathcal{M}_{0}^{1.5}(0.5 * R_{name} + 0.5 * R_{key} + 0.5 * R_{value}) \quad (11)$$

- **Stage 2:** In the second stage, we reduce the λ_{format} by half and add the penalty, maintain the R_{value} as stage 1 and increase the other two parameters to 1.5 times.

$$R = \mathcal{M}_{-1}^{0.5}(R_{format}) + \mathcal{M}_{0}^{3.5}(1.5 * R_{name} + 1.5 * R_{key} + 0.5 * R_{value}) \quad (12)$$

- **Stage 3:** In the last stage, we maintain the λ_{format} as stage 2, increase λ_{value} to 2.5 times its original value and reduce other two parameters.

$$R = \mathcal{M}_{-1}^{0.5}(R_{format}) + \mathcal{M}_{0}^{3.5}(0.5 * R_{name} + 0.5 * R_{key} + 2.5 * R_{value}) \quad (13)$$

Besides, we have recorded the rewards of each subtask of several latest batches to determine which stage to execute, which is the same as the warmup before performing RDS.

4.3 Combination of RDS and TDCL

We combine the two strategies to form our DSCL method. Specifically, we first execute the RDS strategy to obtain the sampling ratios based on the original reward and then execute the TDCL strategy. The overall training framework is illustrated in Figure 2, while the detailed execution process is outlined in Algorithm 1.

Algorithm 1: Dynamic Sampling with Curriculum Learning (DSCL)

Input: Training dataset $\{D, Y\}$, GRPO sampling number G

for $step = 1, ..., step_num$ **do**
  Sample a batch D_b from D ;
  $\widehat{Y} = generate_response(D_b, G)$;
  $R, R_{format}, R_{name}, R_{key}, R_{value} = calc_reward(Y, \widehat{Y})$;
  $Ratio = RDS(R, R_{format}, R_{name}, R_{key}, R_{value})$;
  $R, R_{format}, R_{name}, R_{key}, R_{value} = TDCL(R, R_{format}, R_{name}, R_{key}, R_{value})$;
  $\hat{A}_b = calc_advantage(R)$;
  $\hat{A}_b = \hat{A}_b * Ratio$;
  $update_model(\hat{A}_b)$
end

5 Experiments Settings

5.1 Training Dataset

For the reinforcement learning (RL) phase, we adopt the data composition and processing methodology established by the ToolRL framework (Qian et al., 2025). This approach involves creating a balanced and composite dataset by sampling from three specialized sources (ToolACE (Liu et al., 2024), Hammer (Lin et al., 2024) and xLAM (Zhang et al., 2024)). Following the ToolRL strategy, we sample 2K examples from ToolACE and 1K each from Hammer and xLAM. The rationale behind this curated blend is to expose the model to a comprehensive range of challenges, thereby fostering a robust and generalizable tool-use capability. The composition of our training data and the primary contribution of each source are detailed in Appendix A.1.

5.2 Evaluation Dataset

To ensure a rigorous and directly comparable evaluation, we assess our method on the Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025) and API-Bank (Li et al., 2023), the same benchmarks used in ToolRL (Qian et al., 2025). We do 5 runs for each method and report the average score of the accuracies (%).

5.3 Baselines

To rigorously evaluate the efficacy of our proposed method, we compare it against a set of carefully selected baselines that represent different levels of capability and distinct state-of-the-art philosophies:

- **Qwen2.5-7B-Instruct** (Yang et al., 2024) and **Qwen3-8B** (Yang et al., 2025): Original Qwen models without any additional fine-tuning or RL as the foundation model.
- **Tool-N1** (Zhang et al., 2025): Tool-N1 uses a binary reward for tool learning, evaluating only the final correctness rather than individual sub-tasks.
- **ToolRL** (Qian et al., 2025): An RL framework tailored for tool-use tasks, featuring a fine-grained reward signal that evaluates structural and semantic correctness of tool invocations.
- **DAPO** (Yu et al., 2025): An efficient RL baseline that improves training utility by using Dynamic Sampling to filter uninformative trajectories and Clip-Higher to enhance sample diversity.
- **Sampling by max variance (SMV)** (Xu et al., 2025): A dynamic curriculum that maximizes the reward variance of training batches by selecting a diverse mix of high-reward and low-reward rollouts.
- **Sampling by mean reward (SMR)** (Bae et al., 2025): A dynamic curriculum that uses the model's real-time pass rate (mean reward) for sampling.

Since most of the sampling methods in the baselines have not been used on tool learning task, we implement the results of these methods on tool learning task based on their open-source codes. For SMV and SMR, we perform the method with curriculum learning based on our methodology. The details of the training setting are in Appendix A.2. The details of training prompt are in Appendix C.

6 Experiments

6.1 Main Results

We present the results of our proposed method and other baselines of BFCL and API-Bank in Table 1 and 2. The experimental results demonstrate that both our RDS and TDCL bring a significant improvement over the original ToolRL baseline. The DSCL method, which is combined with the two strategies, achieves the best performance with a 4.16% gain on the BFCL V3 dataset and 3.85% gain on the API-Bank dataset. Notably, our methods also outperform other dynamic sampling approaches such as DAPO, SMV and SMR. This demonstrates that our approach enhances model performance by progressively focusing the training process on more valuable data.

Additionally, we find that it's hard to improve the performance of the model on the tool learning task by directly using the existing dynamic sampling methods such as DAPO. These methods are mainly designed for binary reward problems. Due to the gap in tasks, tool learning does not require an intermediate problem solving process and the response length is relatively short. Thus, some of their techniques, such as the Soft Overlong Punishment of DAPO, are inferior in tool learning. Besides, another reason is that dynamic sampling in tool learning requires warm-up first and cannot start sampling directly, which is analyzed in detail in the Table 3. This result highlights the need for a specially designed dynamic sampling method to address this task.

6.2 Training Analysis

In this section, we will analyze the training process of our method in detail. As shown in Figure 3, we present the reward curves from the baseline ToolRL training compared to the one integrated with our DSCL method. It can be seen that after the introduction of our method, each sub-task can still show an upward trend after the training stabilizes, which demonstrates that our method can continuously provide valuable data to the model through the dynamic sampling strategy. Besides, on the more difficult sub-task, the parameter value task, our method shows a greater improvement compared with the original method, which is due to the fact that our curriculum learning method makes the model pay more attention to this sub-task in the mid-to-late training stage. At the same time, cooperating with the dynamic sampling method to select

Model	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc	Relevance Detection	Irrelevance Detection
Qwen2.5-7B-Instruct (Yang et al., 2024)	47.68	68.98	63.31	8.88	72.22	71.93
Tool-N1 (Zhang et al., 2025)	53.91	77.50	73.39	10.00	77.78	77.85
ToolRL (Qian et al., 2025)	56.96	85.21	73.39	13.25	83.33	75.87
w/ DAPO (Yu et al., 2025)	47.35	68.27	64.55	7.38	77.78	71.80
w/ SMV (Xu et al., 2025)	57.03	85.76	73.79	12.62	83.33	77.59
w/ SMR (Bae et al., 2025)	58.18	86.08	74.87	13.74	83.33	75.43
w/ RDS (Ours)	59.55	85.54	76.10	17.38	83.33	79.86
w/ TDCL (Ours)	59.95	86.25	76.06	18.13	83.33	79.54
w/ DSCL (Ours, RDS+TDCL)	60.25	86.42	76.85	18.50	83.33	78.44
Qwen3-8B (Yang et al., 2025)	57.18	86.08	73.79	15.86	83.33	75.43
Tool-N1 (Zhang et al., 2025)	60.52	88.06	73.43	22.00	**88.89**	71.71
ToolRL (Qian et al., 2025)	62.58	89.54	70.86	30.25	83.33	67.43
w/ DAPO (Yu et al., 2025)	64.07	89.93	75.54	31.13	77.78	82.62
w/ SMV (Xu et al., 2025)	62.96	88.34	72.88	27.13	83.33	75.94
w/ SMR (Bae et al., 2025)	63.47	89.28	72.23	30.00	83.33	76.92
w/ RDS (Ours)	66.21	89.93	77.70	**32.88**	**88.89**	83.07
w/ TDCL (Ours)	65.83	87.15	78.72	31.88	83.33	83.79
w/ DSCL (Ours, RDS+TDCL)	**66.74**	**90.50**	**79.57**	**32.88**	**88.89**	**84.58**

Table 1: Results on the BFCL V3 dataset. The "underline" signifies the better score between the models with the same foundation model. The "**bold**" indicates the best score among all the systems of each language pairs.

Model	Overal Acc	Level 1	Level 2	Level 3
Qwen2.5-7B-Instruct (Yang et al., 2024)	58.29	65.41	43.28	44.27
Tool-N1 (Zhang et al., 2025)	60.47	70.68	46.27	36.64
ToolRL (Qian et al., 2025)	61.81	72.18	56.72	32.82
w/ DAPO (Yu et al., 2025)	58.96	66.17	41.79	45.80
w/ SMV (Xu et al., 2025)	61.64	69.67	49.25	43.51
w/ SMR (Bae et al., 2025)	62.31	72.43	56.72	34.35
w/ RDS (Ours)	63.48	72.18	56.72	40.46
w/ TDCL (Ours)	63.65	74.19	61.19	32.82
w/ DSCL (Ours, RDS+TDCL)	64.99	73.18	65.67	39.69
Qwen3-8B (Yang et al., 2025)	62.48	72.18	56.72	35.88
Tool-N1 (Zhang et al., 2025)	63.32	70.43	55.22	45.80
ToolRL (Qian et al., 2025)	64.32	72.68	62.69	39.69
w/ DAPO (Yu et al., 2025)	66.67	74.19	**64.18**	45.04
w/ SMV (Xu et al., 2025)	65.49	73.43	61.19	43.51
w/ SMR (Bae et al., 2025)	65.66	72.93	62.69	45.04
w/ RDS (Ours)	67.17	74.69	**64.18**	45.80
w/ TDCL (Ours)	67.67	**75.44**	61.19	47.33
w/ DSCL (Ours, RDS+TDCL)	**68.17**	**75.44**	**64.18**	**48.09**

Table 2: Results on the API-Bank dataset.

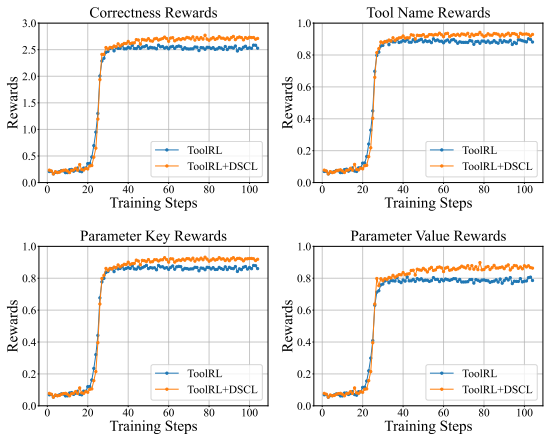

Figure 3: Training rewards of each sub-task. We have mapped the values of each reward to the interval $[0, 1]$ for each sub-tasks. The correctness rewards represent the total reward of the three sub-tasks.

the corresponding data for training for this more difficult sub-task also helps. This result proves the effectiveness of our method.

Additionally, we conduct in-depth experiments on the impact of task difficulty during training in Appendix B. The results prove the necessity of curriculum learning for tool learning task.

6.3 Data Analysis

In this section, we conduct a more in-depth analysis of data distribution on the tool learning task.

We first analyze the model trained with all the data. Figure 4 illustrates the distribution of means and variances of training data with varying difficulty across different training stages and sub-tasks. Specifically, we classify the difficulty of the data based on the number of tools, the number of tool parameters, and the number of dialogue turns, with higher values of these metrics corresponding to increased data complexity. We quantified the counts of easy data and hard data instances as 2293 and 1627, respectively. It can be found that, compared to easy data, the distribution of hard data across each sub-plot exhibits characteristics of lower means and higher variances. Furthermore, this phenomenon becomes more pronounced with the progression of training. Therefore, these sam-

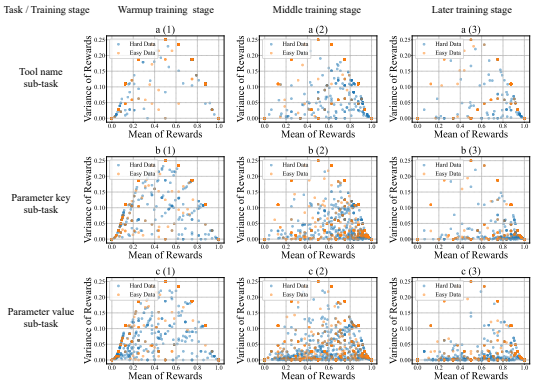

Figure 4: Mean-Variance Scatter Plots of data of different difficulty levels training rewards across different stages of ToolRL training. We have mapped the values of each reward to the interval $[0, 1]$.

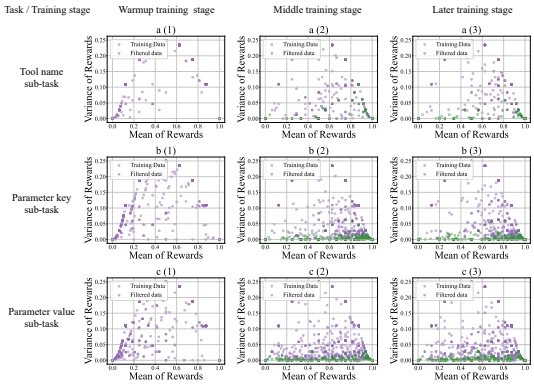

Figure 5: Mean-Variance Scatter Plots of training data (purple points) and filtered data (green points) rewards across different stages of DSCL training. We have mapped the values of each reward to the interval $[0, 1]$.

ples should be given greater emphasis in the later stages of training; otherwise, the model will struggle to overcome its performance bottleneck.

Based on this analysis, we recorded samples that were selected for training versus those that were excluded by our DSCL method for comparison. As illustrated in Figure 5, the majority of the data filtered out by DSCL shows a significant overlap with the simple samples presented in Figure 4. Furthermore, excluding the initial warmup phase where no sampling occurs, we tallied the number of samples used in training versus those discarded in the subsequent two stages. The counts were $(2243, 1397)$ and $(1950, 1690)$, respectively, indicating a gradual decrease in the quantity of data deemed valuable. These comparisons confirm that the DSCL method successfully and continuously focuses on valuable and challenging samples throughout training, thereby guiding the model towards a more optimal set of parameters.

6.4 Training Tips on RDS

Tool learning task requires strict response formatting to extract tool information in a structured manner. As shown in Table 3, directly applying dynamic sampling leads to low reward scores due to formatting errors, because this causes the model to lose access to most training data, resulting in performance comparable to the baseline Qwen2.5-7B-Instruct model. Therefore, we incorporate curriculum learning into our dynamic sampling approach. Specifically, we disable dynamic sampling during the initial warmup stage of training. The RDS method is activated only when the training reward stabilizes, as determined through dynamic monitoring. This curriculum learning strategy enables the model to effectively leverage the benefits of dynamic sampling.

Model	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc
Qwen2.5-7B-Instruct	47.68	68.98	63.31	8.88
ToolRL	56.96	85.21	73.39	13.25
+RDS w/o CL	47.34	68.46	64.02	7.75
+RDS	60.48	87.27	76.63	18.25

Table 3: Ablation Study on the Impact of Curriculum Learning (CL) in RDS. Due to space limitations, we only show the most representative metrics.

7 Conclusion

In this work, we design a Dynamic Sampling method with Curriculum Learning (DSCL) for RL-based tool learning task. Our approach addresses the gaps in existing dynamic sampling methods for tool learning task. The DSCL method contains two strategies, reward-based dynamic sampling (RDS) and task-based dynamic curriculum learning (TDCL). The RDS strategy mainly focuses on the dynamic sampling the training data based on the rewards of all samples generated with the GRPO method. RDS assesses data from three dimensions: the reward mean, the reward variance of a group of rollouts, and the evolution of these statistics over the training history. As for the TDCL, we adaptively prioritize sub-tasks that require more training, ensuring the model focuses on samples with the highest potential for improvement. Our training methodology thoroughly considers the characteristics of the tool learning task, which significantly improve the model performance. Detailed analyses and ablation studies validate our motivation and demonstrate the efficacy of our approach.

8 Limitations

In this work, based on the GRPO reinforcement learning method, we carefully designed a dynamic sampling method for the tool learning task. The experiment demonstrates that our method further improves the effect of reinforcement learning on tool learning. However, our work still has the following limitations:

In order to facilitate alignment with our subsequent ToolRL method, we adopt their dataset settings, including the number and ratio of the training data. After introducing our dynamic sampling method, we can further explore the training data settings that are suitable for our method and further improve its performance.

The current designation of the reward function has been weighted according to the difficulty of different sub-tasks in tool learning. This is similar to the idea of our task-based dynamic curriculum learning method. We can further improve the reward design methods, such as combining the rewards of each sub-task in the same proportion, which can further leverage the advantages of our method and improve performance.

References

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. 2025. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. 2025. Online difficulty filtering for reasoning oriented reinforcement learning. *arXiv preprint arXiv:2504.03380*.

Zhipeng Chen, Yingqian Min, Beichen Zhang, Jie Chen, Jinhao Jiang, Daixuan Cheng, Wayne Xin Zhao, Zheng Liu, Xu Miao, Yang Lu, and 1 others. 2025. An empirical study on eliciting and improving r1-like reasoning models. *arXiv preprint arXiv:2503.04548*.

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. *arXiv preprint arXiv:2503.16219*.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025a. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Zihao Feng, Xiaoxue Wang, Ziwei Bai, Donghang Su, Bowen Wu, Qun Yu, and Baoxun Wang. 2025b. Improving generalization in intent detection: Grpo with reward-based curriculum sampling. *arXiv preprint arXiv:2504.13592*.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18030–18038.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Salman H Khan, and Fahad Shahbaz Khan. 2025. Llm post-training: A deep dive into reasoning large language models. *CoRR*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

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

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, and 1 others. 2024. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. 2025. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, and 1 others. 2024. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, and 1 others. 2024. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D Lee, and Sanjeev Arora. 2025. What makes a reward model a good teacher? an optimization perspective. *arXiv preprint arXiv:2503.15477*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297, New York, NY, USA. Association for Computing Machinery.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, and Willie Neiswanger. 2025. Tina: Tiny reasoning models via lora. *arXiv preprint arXiv:2504.15777*.

Yixuan Even Xu, Yash Savani, Fei Fang, and Zico Kolter. 2025. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

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

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Xingshan Zeng, Weiwen Liu, Xu Huang, Zezhong Wang, Lingzhi Wang, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Ruiming Tang, and 1 others. 2025. Toolace-r: Tool learning with adaptive self-refinement. *arXiv preprint arXiv:2504.01400*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, and 1 others. 2024. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*.

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

A Experiment Settings

A.1 Dataset

In this section, we give the detailed information of our training dataset in Table 4:

- **ToolACE** (Liu et al., 2024): Build a foundational understanding of diverse and complex tool interactions.
- **Hammer** (Lin et al., 2024): Promote deep semantic reasoning over superficial name-matching.

- **xLAM** (Zhang et al., 2024): Develop advanced strategic planning for multi-step, complex tasks.

The details of the evaluation dataset are shown in Table 5 and Table 6 respectively.

BFCL: Analyzing Core Tool Invocation Mechanics. BFCL V3 (Patil et al., 2025) is utilized to measure the fundamental correctness of tool use. Its principal strength is a diagnostic, multi-faceted structure that deconstructs a single tool use into granular dimensions, as detailed in Table 5. This approach is critical for isolating specific failure modes—such as distinguishing an error in tool selection from one in parameter formatting—thereby enabling a precise, component-level analysis of our model's fidelity.

API-Bank: Evaluating Multi-Step Planning and Reasoning. Complementing BFCL's focus on mechanics, the API-Bank benchmark(Li et al., 2023) evaluates higher-order reasoning within realistic, multi-turn dialogues. It employs a hierarchical evaluation that tests progressively complex skills, from basic tool execution to autonomous, multi-step planning, with a full breakdown provided in Table 6. The benchmark's foundation on manually annotated, authentic API interactions allows us to rigorously validate our model's improvements in strategic tool use.

A.2 hyperparameter

For our training process, we mainly follow the setting of ToolRL, all the modifications and parameters of our method are shown in the Table 7:

All the models are trained with GRPO method using the veRL (Sheng et al., 2025) framework and conducted on 8 H20 GPUs with 96GB of RAM for 15 epochs.

B Tool Complexity and Task Difficulty Analysis

We analyze how task complexity factors affect training performance across epochs. Specifically, we examine two complexity dimensions: (1) the number of available tool APIs and (2) the number of tool parameters. Our analysis covers four task categories: the overall task, tool name selection, parameter key extraction, and parameter value completion.

For mean reward, both complexity factors show consistent trends across all task categories. As the number of tool APIs or tool parameters increases, mean rewards consistently decrease during training (Figure 6). This suggests that increased tool complexity directly reduces task performance.

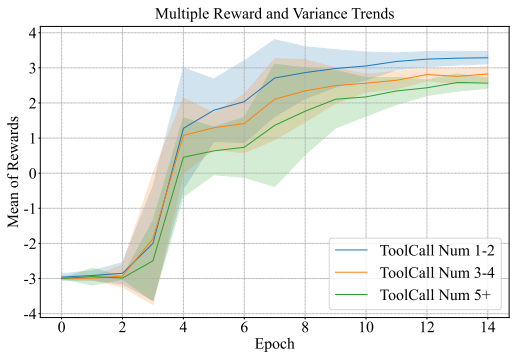

Figure 6: The relationship between training epochs and overall mean reward (the lines) and their corresponding variances (the shaded area) in ToolRL, categorized by the number of tool APIs.

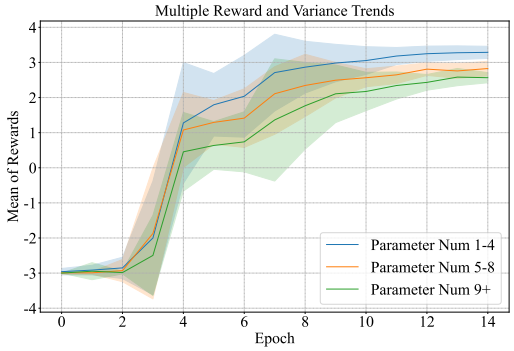

Figure 7: The relationship between training epochs and overall mean reward (the lines) and their corresponding variances (the shaded area) in ToolRL, categorized by the number of tools' parameters.

For the variance of reward, the opposite trend emerges. As the number of tool APIs or tool parameters increases, the reward variance increases consistently across all categories (Figure 7). This indicates that complex tool environments introduce greater performance instability and learning difficulty.

Together, these findings demonstrate that task difficulty increases proportionally with both the size of the available tool set and the complexity of individual tool parameters.

C Prompts

We follow the prompt designation of ToolRL (Qian et al., 2025). The specific system prompt and user prompt design are shown in Table 8 and Table 9.

Dataset	Key Contribution	Samples
ToolACE	**Diversity & Complexity**: Covers single, parallel, dependent, and non-tool-use scenarios to teach *when* and *how* to use tools.	2,000
Hammer (Masked)	**Semantic Reasoning**: Uses function masking to force the model to understand API descriptions instead of relying on names.	1,000
xLAM	**Strategic Planning**: Features complex tasks requiring orchestration of multiple tools and management of dependencies.	1,000

Table 4: RL Training Dataset Composition and Rationale. We adopt the data strategy from ToolRL, combining three specialized datasets for a total of 4,000 training samples. Each dataset is chosen to foster a specific capability: ToolACE for foundational diversity, Hammer for semantic reasoning, and xLAM for advanced strategic planning.

Main Category	Subtotal	Evaluation Dimension	Count	Evaluation Purpose
Non-Live (Single-Turn)	**1,390**	Non-Live AST	1,150	To test for **syntactic correctness**.
		Irrelevance Detection (Static)	240	To test for **hallucination avoidance**.
Live (Single-Turn)	**2,251**	Live AST	1,351	To test for **executable correctness**.
		Relevance Detection (Live)	18	To test for **correct tool selection**.
		Irrelevance Detection (Live)	882	To test for **hallucination avoidance in a live context**.
Multi-Turn	**800**	Multi-Turn Dialogue	800	To test for **stateful and sequential reasoning**.

Table 5: A Visually Enhanced Breakdown of the BFCL Benchmark (Revised Layout)

Statistic	Count
Domains	8
APIs	73
Dialogues	314
Turns	914
Single-Call Turns	363
Multi-Call Turns	122
Call Tasks	214
Retrieve+Call Tasks	50
Plan+Retrieve+Call Tasks	50
Avg. Turns per Dialogue	2.91

Table 6: Statistics of the API-Bank Evaluation Set.

Hyperparameter	Value
Modifications:	
Number of Rollouts	8
Our Method:	
t_{mean}	0.5
t_{var}	0.1

Table 7: Statistics of the API-Bank Evaluation Set.

You are a helpful multi-turn dialogue assistant capable of leveraging tool calls to solve user tasks and provide structured chat responses.

Available Tools
In your response, you can use the following tools:
1. Name: getGastroenterologyReport
Description: Retrieve gastroenterology report for a patient
Parameters: {"patient_id": {"description": "The unique identifier of the patient", "type": "string", "default": ""}, ...}
2. ...

Steps for Each Turn
1. **Think:** Recall relevant context and analyze the current user goal.
2. **Decide on Tool Usage:** If a tool is needed, specify the tool and its parameters.
3. **Respond Appropriately:** If a response is needed, generate one while maintaining consistency across user queries.

Output Format
<think> Your thoughts and reasoning </think>
<tool_call>
{"name": "Tool name", "parameters": {"Parameter name": "Parameter content", "... ...": "... ..."}}
{"name": "... ...", "parameters": {"... ...": "... ...", "... ...": "... ..."}}
...
</tool_call>
<response> AI's final response </response>

Important Notes
1. You must always include the '<think>' field to outline your reasoning. Provide at least one of '<tool_call>' or '<response>'. Decide whether to use '<tool_call>' (possibly multiple times), '<response>', or both.
2. You can invoke multiple tool calls simultaneously in the '<tool_call>' fields. Each tool call should be a JSON object with a "name" field and an "parameters" field containing a dictionary of parameters. If no parameters are needed, leave the "parameters" field an empty dictionary.
3. Refer to the previous dialogue records in the history, including the user's queries, previous '<tool_call>', '<response>', and any tool feedback noted as '<obs>' (if exists).

Table 8: System prompt

Dialogue History
<user> {{ Initial User Input }} </user>

<think> Round 1 Model Thought </think>
{{ Round 1 model output <tool_call> or <response> }}
<obs> Round 1 Observation </obs>
... ...

<user> {{ User Input }} </user>
... ...

Table 9: User prompt