Democratizing Tool Learning with Environments Fully Simulated by a Free 8B Language Model

Chenming Tang[♠♣] **Hsiu-Yuan Huang**[♠♣◇]
Weijie Liu[◇†] **Junqiang Zheng**[◇] **Saiyong Yang**[◇] **Yunfang Wu**[♠♣†]
[♠]National Key Laboratory for Multimedia Information Processing, Peking University
[♣]School of Computer Science, Peking University [◇]LLM Department, Tencent
tangchenming@stu.pku.edu.cn jagerliu@tencent.com wuyf@pku.edu.cn

Abstract

Reinforcement learning has become a prevalent paradigm for training tool calling agents, which typically requires online interactive environments. Existing approaches either rely on training data with ground truth annotations or require advanced proprietary language models (LMs) to synthesize environments that remain fixed once created. In this work, we propose TRUSTEE, a cost-friendly method for training tool calling agents with dynamic environments fully simulated by open-source LMs that can be as small as 8B, including task generation, user simulation, tool simulation, and trajectory evaluation, paired with an adaptive curriculum learning mechanism that controls task difficulty dynamically. Our empirical results show that TRUSTEE outperforms baselines which require extra external resources in most cases. These confirm that, with a sufficiently sophisticated design, even simulated environments with a local 8B LM as the backbone could set a strong baseline for tool learning. We hope our proposed paradigm could democratize tool learning and inspire future research on environment scaling with limited resources.

1 Introduction

Tool calling, a.k.a. function calling, enables language models (LMs) to interact with the external environment and act as autonomous agents [1, 2, 3]. Earlier work teaching LMs to call tools (*i.e.*, tool learning) typically adopts supervised fine-tuning (SFT) [4, 5, 6] or single-turn reinforcement learning from verifiable rewards (RLVR) [7, 8, 9], relying on annotated data. With the prevalence of agentic reinforcement learning (agentic RL) [10], recent research has been focusing on tool learning via multi-turn RL. However, due to the cost and limited availability of real-world users and tool infrastructures, there is a significant challenge in scaling the environment[2], as this form of RL requires a large number of online interactions including calling many tools and dynamically interacting with the user, which also previously challenged evaluation [11, 12] and SFT data construction [13, 14, 15].

One approach to environment scaling is to apply advanced LMs for environment simulation, but most of these works only partially simulate the environment (either the tools [16, 17, 18] or the users [19]) and rely on existing data or environments (*i.e.*, supports some specific tasks only). For instance, Simia-RL [20] simulates the responses of tools with LMs during interactions, but requires established data with annotated training tasks as the input (*e.g.*, τ-bench [21]) and does not support the multi-turn interaction between the agent and the user. Another approach is to make use of synthesized code and databases [22, 23], but such an approach requires a complex generation pipeline for high-quality synthesis using expensive proprietary LMs and the generated environments and tasks are fixed once created, which may be too easy for strong agents but overly difficult for weak ones.

[†]Corresponding authors.

[2]In this paper, *environment* refers to all the external components for the agent, including users, tools, verifiers, *etc.*

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To this end, we introduce TRUSTEE, **T**ool lea**R**ning **U**sing **S**imula**TE**d **E**nvironments, a cost-friendly method for training tool calling agents with a tool repository and an open-source LM as the only resources required. TRUSTEE uses an LM to generate the task, simulate the user, simulate tool responses, and evaluate the trajectories, which provides complete environments for tool learning while independent of any existing training data or tasks, human annotations or interactions, realistic executable tools, or verifiable environments engineered by experts or proprietary LMs. Moreover, we design an adaptive curriculum learning mechanism that takes advantage of the dynamically controllable nature of the simulated environments and tasks, which controls the difficulty of the tasks in various dimensions adaptively and ensures a smooth learning process for the tool calling agent.

Our experiments validate the effectiveness of TRUSTEE. For QWEN3-8B-BASE and QWEN3-4B, TRUSTEE brings more positive gains than baselines in most cases. Moreover, the QWEN3-8B LM, when trained solely with environments simulated by itself, generally has better and more balanced performance than baselines which either require annotated data or need proprietary LMs to engineer the environments. This demonstrates that, with a sufficiently sophisticated design, simulated environments may be a powerful approach to tool learning that can even beat realistic data or synthesized verifiable environments. Analysis from different perspectives also helps us better understand the simulation quality, training stability, contributions of components, and cost efficiency.

Our contributions include:

- We introduce TRUSTEE, a paradigm of generating tasks and fully simulating environments with an LM for online tool learning.
- We propose adaptive curriculum learning for tool learning, which dynamically keeps the tasks and environments at an appropriate difficulty for the policy.
- We empirically validate the effectiveness of TRUSTEE backboned by a free 8B LM, demonstrating tool learning does not necessarily require established data, executable tools, or verifiable environments from human experts or advanced proprietary LMs, moving a step towards democratized tool learning.

Our code is available at `https://github.com/JamyDon/TRUSTEE`.

2 Method

2.1 Problem Formulation

We formalize the tool calling agent task as a partially observable Markov decision process [24], defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{O} \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A} = \mathcal{A}_{\text{text}} \cup \mathcal{A}_{\text{tool}}$ is the action space (textual response or tool calling), $\mathcal{P}$ is the state transition probability, $\mathcal{R}$ is the reward space, γ is the discount factor, and $\mathcal{O}$ is the observation space. At the t-th POMDP step, the agent driven by an LM θ receives an observation o_t based on the state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$ based on o_t and its policy π_θ, receives a reward $r_t = \mathcal{R}(s_t, a_t)$, and the environment transitions per $\mathcal{P}(s_{t+1} \mid s_t, a_t)$. Instead of the standard discounted return, we solely use the final reward of each trajectory τ as the objective (see Appendix F.2 for details): $\mathcal{J}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[r_\tau]$.

2.2 Fully Simulated Environment

Our fully simulated environment is demonstrated in Figure 1, which is backboned by an LM Φ. There are three stages for each training iteration[3]. The first generates tasks for this batch (Section 2.2.1). The second performs simulated interactions to collect trajectories (Section 2.2.2). The last rewards the trajectories and optimizes the policy (Section 2.2.3). An algorithm description of TRUSTEE is in Appendix A, and the prompt templates for the backbone LM can be found in Appendix B.

2.2.1 Task Generation

At the n-th training iteration, for each task $\mathcal{T}_i$, we first sample (denoted by ζ) a subset f of a large repository of tools $\mathcal{F}$, with the size of the subset determined by the task difficulty $\mathcal{D}_n$ (see Section 2.3): $f = \zeta(\mathcal{F}, \mathcal{D}_n)$. In practice, the tool repository can be from various sources like existing datasets,

[3] In this section, we use the term "iteration" instead of "step" for training to distinguish from POMDP/interaction steps.

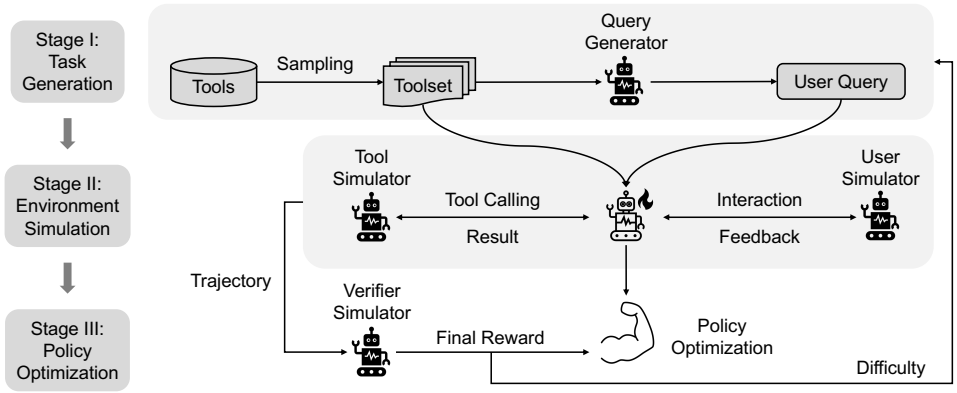


Figure 1: The overview of TRUSTEE.

crawled tools from the Internet, manually designed tool sets, *etc.* Note that the tools only consist of basic textual information (names and parameters) and do not have to be actually executable functions.

Then, we perform task generation based on the backbone LM:

$$e_0 \sim \mathcal{P}_\Phi(\cdot|\Omega_G), \tag{1}$$

where $\Omega_G = \Omega(f, \mathcal{D}_n)$ is the structural prompt for task generation based on the tools f and difficulty $\mathcal{D}_n$, and $e_0 = (p, g, q, c)$ is the generated task including the user persona p, the user intent g (*i.e.*, the ultimate goal of the task), the first user query to trigger an interaction q, and the expected tools c to be called to complete the task. Finally, after determining the agent's system prompt h per $\mathcal{D}_n$ (see Section 2.3.1), we have the initial state s_0 based on all these information: $s_0 = (f, p, g, q, c, h, \mathcal{D}_n)$.

2.2.2 Simulated Interaction

We perform simulated interaction based on the generated tasks. The interaction environment includes a tool executor and a user, both simulated by the backbone LM Φ.

For each task $\mathcal{T}_i$, we sample M interaction trajectories. For each trajectory τ, the policy agent π_θ begins the interaction after receiving the first observation $o_0 = \tau = (h, f, q)$, including the system prompt h, available tools f, and the first user query q. It interacts with both the tool executor and the user by generating actions based on the current trajectory τ (which is exactly the current observation):

$$a_t \sim \pi_\theta(\cdot|\tau), \tag{2}$$

which will be appended to the trajectory: $\tau = \tau \oplus a_t$.

If a_t is a tool call, the simulated tool will be called to generate a tool result based on the structural tool simulation prompt $\Omega_T = \Omega(f, g)$ which includes the tools f and the user intent g. Otherwise, if a_t is a textual response to the user, the simulated user will be called to generate a user message based on the structural user simulation prompt $\Omega_U = \Omega(f, g, p)$ which includes the available tools f, the user intent g, and the user persona p. The feedback is generated based on the corresponding prompt:

$$e_t \sim \mathcal{P}_\Phi(\cdot|\tau, \Omega_{T/U}), \tag{3}$$

which will then be appended to the trajectory: $\tau = \tau \oplus e_t$.

The user simulator may choose to end the task by generating an empty response if it believes there is no need to perform more interactions. Otherwise, the interaction will continue until the sequence length or the interaction turn reaches a pre-defined limit.

Regarding the simulator, we do not require it to perfectly reproduce the real world. Reasonable and consistent interaction feedback is enough. For example, when learning to call a weather-search tool, whether the simulated API returns `Seattle, April 18: light rain, 52 °F` or the actual weather is different is less important than whether the agent learns to choose the right tool, pass the right city and date arguments, read the returned fields, and answer consistently. This view is also supported by ZeroSearch [25], which trains search agents with simulated searches and reports that a 7B retrieval module achieves comparable performance to real search, while a 14B one can even surpass it. Therefore, in case the simulator is not an advanced LM (or even identical to the policy LM), it may be viewed as a limit on the training signal source, but not necessarily on the final policy.

2.2.3 Rewarding and Optimization

For each sampled trajectory τ, when the interaction is finished, a verifier simulated by the backbone LM Φ is called to evaluate the whole trajectory and generates a final reward with the structural verifier simulation prompt $\Omega_V = \Omega(f, g, p, c, \mathcal{D}_n)$ based on the available tools f, the user intent g, the user persona p, the expected tools c, and the task difficulty $\mathcal{D}_n$:

$$r \sim \mathcal{R}_\Phi(\cdot|\tau, \Omega_V). \tag{4}$$

In this work, we have $r \in \{-1, 0, 1\}$ in our implementation. The trajectory receives a positive reward if all the criteria are met, a neutral reward if most criteria are met, and a negative reward if there are severe errors or the criteria are scarcely met.

We use GRPO [26, 10] for optimization due to its stability and simplicity. Details are in Appendix D.

2.3 Adaptive Curriculum Learning

An advantage of the fully simulated environment is that every component is dynamic and adjustable in an online manner, which naturally enables adaptive curriculum learning which keeps a moderate difficulty for the current policy and ensures good learnability.

2.3.1 Task Difficulty

The task difficulty involves the following dimensions. More details can be found in Appendix E.1.

Number of Tools The number of tools sampled from $\mathcal{F}$ and the number of expected tool calls for task generation directly determine the overall difficulty. The more available tools and expected calls, the harder it is for the agent to make use of tools properly and finish the task precisely.

Number of Interaction Turns If a task requires more interaction turns, it is naturally more complex. Therefore, we use the number of expected turns, which will be passed to the LM Φ for task generation, to control the difficulty. Meanwhile, the maximum number of interaction turns also evolves dynamically with the expected turns to ensure efficient training.

System Prompt We expect the agent to learn to perform high-quality tool calling under a general system prompt and make the specificity of the system prompt gradually decrease during training. To this end, we design the system prompt h at different degrees of specificity. The most specific one describes the overall multi-turn tool calling task in detail with hands-on instructions (a few paragraphs). Then, a more concise one presents the core notes (a few sentences) regarding the agentic task only. After that, a highly consolidated one states the task in a sentence (*e.g.*, "You are a helpful multi-turn dialogue assistant capable of leveraging tool calls to solve user tasks."). Finally, the hardest system prompt is the most general one (*e.g.*, "You are a helpful assistant."), with no special descriptions regarding any specific task.

User Typically, a less expert user makes the task harder due to a lower level of knowledge [27], and an ambiguous user query usually confuses the agent and requires it to ask for further clarification. Therefore, we design three kinds of user personas (expert, beginner, and novice) together with three ambiguity degrees (clear, somewhat ambiguous, and highly ambiguous). Then, we use this as an input to the LM Φ for task generation which generates a more specific description of the user persona p (which will be further passed to the user simulator) and a user query q based on the requirements.

Rewarding Criteria Stricter evaluation criteria naturally make it harder for the agent to obtain a high reward. We increase the difficulty by gradually taking more criteria into consideration when calling the verifier Φ, which gives a positive reward only when all the criteria are met. Concretely, we design five levels of criteria: meeting user intent, using expected tools correctly, no hallucination, no redundant trials, and efficiency. These draw a step-by-step path to a good tool calling agent.

2.3.2 Difficulty Evolution

The difficulty is updated after each training iteration based on the overall reward of the finished batch. If the averaged reward $\bar{r}_n$ of training iteration n is higher than a threshold η_{high}, the difficulty level

for the next iteration $\mathcal{D}_{n+1}$ will increase by a step size δ. Vice versa.

$$\mathcal{D}_{n+1} = \begin{cases} \mathcal{D}_n + \delta & \text{if } \bar{r}_n > \eta_{\text{high}} \\ \mathcal{D}_n - \delta & \text{if } \bar{r}_n < \eta_{\text{low}} \\ \mathcal{D}_n & \text{otherwise} \end{cases}. \tag{5}$$

Then, all of the dimensions in Section 2.3.1 will be derived based on $\mathcal{D}_{n+1}$. In this work, we design a large value range (*e.g.*, from 1 to 100) for $\mathcal{D}$ and convert it to the scales of different dimensions. For example, since the range of user persona is 0-2 (0 for expert, 1 for beginner, and 2 for novice), the derived user persona level will be $\lfloor \mathcal{D}_{n+1} \times 0.02 \rfloor$. In this way, the user persona will be expert for difficulty from 1 to 49, beginner for difficulty from 50 to 99, and novice for difficulty 100. Since this mapping depends on the stepped design of each dimension, it is not treated as a hyper-parameter.

Since the difficulty works for every task in a batch, the batch may converge to a specific pattern of tasks, which hurts the diversity during training. For example, a detailed system prompt is always paired with an expert user. To this end, we employ a soft design of curriculum, which does not always follow the current difficulty strictly. To be specific, for each task, there is a probability of ϵ to sample the tool number, the expected tool callings, the interaction turns, the user persona, and the query ambiguity randomly within the current difficulty. In this way, the task patterns can be more diverse.

2.4 Source of Improvement

The essential engine of learning in TRUSTEE lies in the gap between the agent's limited observations and the environment's privileged knowledge (*i.e.*, information asymmetry).

Lemma 2.1 (Information Asymmetry)**.** *The environment operates on a broader information set $\mathcal{I}_{env} = \{\tau, g, p, c, \mathcal{D}_n\}$ while the agent operates solely on $\mathcal{I}_{agent} = \{\tau\}$. Therefore, the environment may provide a training signal based on its privilege knowledge unknown to the agent.*

Consequently, the improvement of TRUSTEE goes beyond a simple teacher-student distillation (from Φ to π_θ) but can come from the structural design of the environment.

Proposition 2.1 (Structural Improvement)**.** *Let $J(\pi) = \mathbb{E}_{\tau\sim\pi}[R(\tau)]$ denote the expected return of policy π. Given the environment backbone Φ, the environment scaffolding $\mathcal{E}$ (including task generation, interaction, and rewarding), and the structural prompt set Ω for the backbone, the improvement $\Delta J = J(\pi_\theta) - J(\pi_0)$ can be decomposed into the transfer gain (from the backbone Φ's margin) and the structural gain (from the structural design including $\mathcal{E}$ and Ω):*

$$\Delta J = \textit{Gain}_{\textit{trans}}(\Phi, \pi_\theta) + \textit{Gain}_{\textit{struct}}(\mathcal{E}, \Omega), \tag{6}$$

where $\textit{Gain}_{\textit{struct}} > 0$ even when Φ is not superior to θ.

The detailed justifications for this proposition are provided in Appendix C.

3 Experimental Setup

3.1 Data

Evaluation Benchmarks We evaluate baselines and our method on the Berkeley Function Calling Leaderboard (BFCL) V4 [28] (we focus on two single-turn subsets "Non-Live" and "Live", and one multi-turn subset "Multi-Turn") and τ^2-bench [27]. The former tests general tool calling and the latter evaluates interactive tool calling in three realistic scenarios ("Airline", "Retail", and "Telecom"). Due to a limited budget, in our analytical experiments, we evaluate with BFCL V4 only.

Tool Repository We instantiate our tool repository $\mathcal{F}$ using the tools collected by ToolBench [6], which are representational state transfer (REST) APIs gathered from RapidAPI[4] with detailed descriptions, spanning 49 diverse categories like social media, e-commerce, and weather. Details of data preprocessing are in Appendix E.2. Experiments on stateful tools are in Appendix F.4.

[4] https://rapidapi.com/hub

3.2 Language Models

We focus on the QWEN3 [29] herd of language models, which is widely used for reinforcement learning in the community. To be specific, we experiment with QWEN3-8B-BASE, QWEN3-4B, and QWEN3-8B as the agent LM θ. Throughout our experiments, we use QWEN3-8B as the environment backbone LM Φ (see the discussion regarding the backbone LM in Appendix F.3).

3.3 Baselines and Methods

Our evaluated baselines and methods include: (1) **Original Model**: the agent LM before training. (2) **ToolRL** [7]: a single-turn RLVR method using existing SFT data [30, 15, 31] with ground truth annotations. (3) **Tool-N1** [8]: another single-turn RLVR method using existing SFT data [30, 15]. (4) **EnvScaler** [22]: a method that generates environments via programmatic synthesis using proprietary LMs. (5) **AWM** [23]: another method that employs programmatic environment synthesis using proprietary LMs. (6) **Simia-RL** [20]: a method simulating the tools with proprietary LMs, requiring benchmark-specific environments. (7) **TRUSTEE (ours)**: our method training the agent in environments simulated by an 8B open-source LM.

3.4 Implementation Details

We use a single node with 8× NVIDIA H20 GPUs. For TRUSTEE, 4 GPUs are used for agent training and the other 4 GPUs are used for deploying the simulation backbone LM; for other baselines, all the 8 GPUs are used for agent training. Since no extra hardware beyond the common budget is given to TRUSTEE, the cost is fair at the node level. Our training script is developed on the basis of `verl` [32]. The simulation backbone LM is deployed using `vLLM` [33]. We extend the backbone LM's initial context size of 32,768 to 131,072 with a 4× YaRN [34] factor to avoid overlong trajectories.

For TRUSTEE, the step size of difficulty evolution δ is 3. The probability of soft curriculum ϵ is 0.5. The reward thresholds for difficulty update η_{low} and η_{high} are 0.0 and 0.5, respectively. The maximum numbers of tokens generated by the simulated user, tool executor, and verifier are all 2,048. The environment backbone LM operates with a frequency penalty of 0.1. The range of the number of tools is 1-10. The range of the expected tool callings is 1-3. The ranges of the expected turns, maximum user turns, maximum tool turns, maximum agent turns are 1-2, 1-3, 1-3, 1-6, respectively. To ensure the sampled tools from $\mathcal{F}$ are not overly discursive, we prioritize tools from the same category when sampling. We begin to sample from a new category only when all the tools have been selected in the previous one. To ensure stable tool learning, we include an extra format check before rewarding, in which the agent receives a negative reward if no tools are called throughout the interaction.

The details of evaluation can be found in Appendix E.3. The details of baseline methods are presented in Appendix E.4. The hyper-parameters for training are listed in Appendix E.5.

4 Results and Analysis

4.1 Main Results

The main end-to-end results are presented in Table 1.

For the pre-trained QWEN3-8B-BASE, TRUSTEE brings significant improvements and secures the highest scores in 5 out of 6 subsets. It improves the base model by **10**× and **15**× on the two single-turn subsets of BFCL. On BFCL's multi-turn tool calling subset, it even improves the performance from **0.00** to **7.88** and outperforms the second highest baseline by **13**×. On τ^2-bench, it makes consistent improvements in all the subsets, with a significant **3.5**× improvement on "Telecom". On the contrary, other baselines provide relatively limited gains, especially on the multi-turn subset of BFCL, where all the baselines fail to achieve meaningful progress.

The post-trained QWEN3-4B and QWEN3-8B already have a good capacity in tool calling, but TRUSTEE still brings generally positive gains in most cases. For QWEN3-4B, TRUSTEE performs generally better than all the baselines except EnvScaler, which requires a lot more time in training due to the complex environments (see Section 4.5). Nevertheless, for QWEN3-4B, no method provides consistent gains in all the cases, and we acknowledge TRUSTEE fails to improve on BFCL's Live and τ^2-bench's Airline, indicating it is still challenging to further improve on an already post-trained LM.

Table 1: Main results. The highest results for each LM are **bolded**. "DF" refers to data-free (*i.e.*, no established data) while "OP" indicates using open-source LMs only. The BFCL results of methods with "†" are evaluated under the direct prompting setting instead of the standard native function calling (FC) setting because they score nearly 0 on all the categories under the default FC setting.

Method	DF	OP	BFCL V4			τ^2-bench		
			Non-Live	Live	Multi-Turn	Airline	Retail	Telecom
QWEN3-8B-BASE	-	-	8.38	4.74	0.00	26.00	4.39	1.75
ToolRL	✗	✓	40.83	56.33	0.25	30.00	**6.14**	5.26
Tool-N1†	✗	✓	37.85	29.53	0.12	**32.00**	4.39	4.39
EnvScaler†	✓	✗	75.29	62.03	0.62	28.00	**6.14**	0.00
TRUSTEE	✓	✓	**83.92**	**72.54**	**7.88**	**32.00**	5.26	**6.14**
QWEN3-4B	-	-	87.77	82.16	32.12	22.00	27.19	9.65
ToolRL	✗	✓	79.44	71.65	13.00	2.00	11.40	0.00
Tool-N1	✗	✓	88.44	77.28	34.12	24.00	9.65	11.40
AWM	✓	✗	87.12	80.61	30.00	20.00	22.81	9.65
EnvScaler	✓	✗	86.50	**82.24**	**38.75**	**28.00**	**29.82**	10.53
TRUSTEE	✓	✓	**88.96**	81.35	35.62	22.00	28.07	**16.67**
QWEN3-8B	-	-	87.77	80.31	39.88	24.00	42.98	15.79
ToolRL	✗	✓	88.77	81.20	41.25	22.00	16.67	14.04
Tool-N1	✗	✓	87.83	78.90	44.38	28.00	36.84	10.53
AWM	✓	✗	87.52	80.46	39.75	30.00	36.84	12.28
EnvScaler	✓	✗	87.42	81.20	44.50	24.00	42.11	8.77
Simia-RL	✗	✗	32.40	46.20	2.00	**38.00**	35.09	3.51
TRUSTEE	✓	✓	**89.65**	**82.16**	**46.38**	26.00	**45.61**	**17.54**

For QWEN3-8B, TRUSTEE improves on all the subsets and secures the highest scores in 5 out of 6 cases, generally outperforming other baselines. Note that, for QWEN3-8B, the agent and the environment backbone share the same foundation LM. This demonstrates that with TRUSTEE, QWEN3-8B even benefits from itself when no annotated data or advanced LMs are involved, which empirically confirms the structural gain discussed in Section 2.4.

4.2 Analysis of the Simulated Environment

Besides end-to-end performance, we also evaluate the overall reliability of TRUSTEE's simulated environment. We sample 100 trajectories via QWEN3-8B self-play in the simulated environment, and call DEEPSEEK-V4-PRO [35], an advanced LM judge, to score (from 1 to 5) the simulation quality from various aspects. As shown in Table 2, all the scores are above 3.5, suggesting a healthy simulation quality. The quality is especially good in task generation and user simulation, with scores over 4.5. As discussed in Section 2.2.2, we do not need a perfect simulator, and the current quality is enough.

Table 2: Simulation quality.

Aspect	Score
Task Generation	4.54
Tool Simulation	3.77
User Simulation	4.64
Final Reward	3.90
Internal Consistency	4.37
Overall Fidelity	3.88

We also specifically evaluate the reliability of the verifier simulator. We use the test split of ToolRL's dataset [7], in which each instance has a gold tool call and a rule-based scorer. We collect 410 trajectories whose correctness is mostly known by construction: gold positives, perturbations of gold calls inducing errors, and additional trajectories generated by QWEN3-8B labelled by the scorer. Then, we run QWEN3-8B as a reward verifier, and find it separates trajectories well. Against the rule-based scorer, the Spearman correlation is 0.70, accuracy is 0.87, F1 is 0.81 (precision is 0.72, recall is 0.93), and ROC-AUC is 0.89. The mean reward is 0.93 for correct and -0.08 for erroneous trajectories.

Finally, we perform manual audit on the only 20 trajectories generated by QWEN3-8B that were judged incorrect by the rule-based scorer, and find that half of them are actually correct. Some (seven) are due to the overly strict exact-match scorer. Others (three) are because the gold annotation is itself malformed. This shows that rule-based labels can be brittle in certain cases. Meanwhile, QWEN3-8B verifier has an accuracy of 0.85 on the 20 manually audited trajectories if we read a reward of +1 as good and 0 or -1 as bad. Therefore, the simulated verifier is generally reliable.

4.3 Training Dynamics and Stability

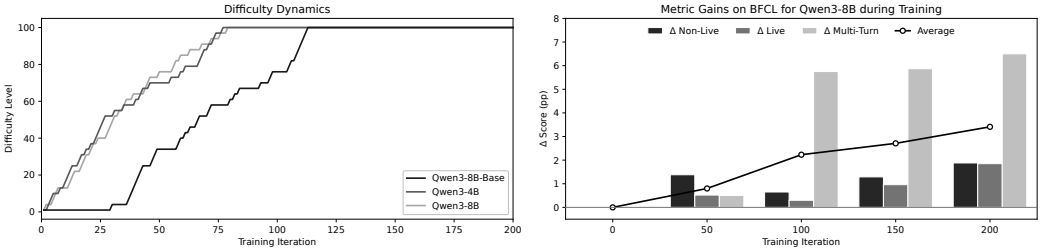


Figure 2: The dynamics of difficulty level (top) and metric gains (bottom).

Figure 2 presents the dynamics of difficulty level and BFCL V4 metric gains. Throughout the training, the curriculum difficulty increases smoothly and reaches the upper bound around half of the training, and the overall performance is improving as the training proceeds. Meanwhile, according to our observation, the gradient norms remain bounded, the KL grows smoothly while remaining moderate, and the entropy decreases gradually. These suggest stable training throughout the process.

4.4 Ablation Study

We perform ablation studies on the environment's information set, adaptive curriculum learning, and hyper-parameters. The results on BFCL V4 with QWEN3-8B are in Table 3. These studies serve as stress tests of deliberately weakened variants, aiming to verify whether each component is necessary and understand the effect of hyper-parameters. It is therefore expected that the resulting method becomes less effective and can even hurt the original model, especially when the original QWEN3-8B model has already been post-trained. This supports the indispensability of these components and the importance of moderate hyper-parameters.

Table 3: Ablation results on BFCL (QWEN3-8B).

Setting	**Non-Live**	**Live**	**Multi-Turn**
QWEN3-8B	87.77	80.31	39.88
TRUSTEE default	**89.65**	**82.16**	**46.38**
(I) Environment's Information Set			
w/o intent	88.42	80.90	43.00
w/o persona	86.73	79.20	43.12
w/o expected calls	88.42	79.94	44.38
(II) Curriculum Learning			
w/o curriculum learning	88.25	79.64	41.50
w/o soft curriculum learning	88.31	80.98	44.50
w/o tool	87.31	78.46	27.38
w/o turn	87.85	81.79	41.75
w/o system prompt	87.56	80.46	40.75
w/o user	87.54	79.57	32.75
w/o criteria	88.42	81.05	43.12
(III) Hyper-parameters			
Number of tools ↑	87.63	80.46	41.25
Number of tools ↓	87.35	79.79	40.25
Number of turns ↑	87.87	80.53	44.25
Number of turns ↓	87.98	80.09	35.88
Step size ↑	88.06	79.42	38.75
Step size ↓	89.58	81.42	43.62

Ablation Study on the Environment's Information Set We analyze the three components of information used by the environment backbone LM: user intent g, user persona p, and expected tool calls c. We ablate these components by making them invisible to the backbone LM Φ for user, tool, and verifier simulation. As shown in Table 3 (Part I), removing any one of the components hurts the performance on all the subsets. Without user intent, the environment backbone does not have sufficient information regarding the latent ground truth. Without user persona, the user simulator may not behave consistently and match the difficulty. Without expected calls, the verifier may deviate from the initial design of the task. These suggest the indispensability of all three components.

Ablation Study regarding Adaptive Curriculum Learning First, we ablate the whole adaptive curriculum learning module by fixing the difficulty at the highest level. Without a gradual learning process, the overall performance downgrades, especially on BFCL's Multi-Turn subset, where the

scores decrease by **4.9** points for QWEN3-8B, shown by Table 3 (Part II). Then, we ablate the soft curriculum design described in Section 2.3.2, and the corresponding results confirm that a randomized difficulty prevents the agent LM from overfitting to undiversified tasks and environments and yields better performance. Finally, we perform an ablation study on the various dimensions for difficulty control: number of tools, number of interaction turns, system prompt, user, and rewarding criteria. We ablate each component by fixing it at the highest difficulty level throughout the training. As the results show, all five dimensions contribute positively on all three subsets of BFCL, which confirms all five dimensions are necessary to ensure a successful training outcome.

Effect of Hyper-parameters We analyze the effect of hyper-parameters in our method, including maximum number of available tools, maximum interaction turns, and the step size δ of difficulty evolution. We set three levels for each hyper-parameter: lower ($\downarrow$), default, and higher ($\uparrow$), with default as the value used in our main experiments. The detailed hyper-parameter values for each setting are presented in Appendix E.6. As shown in Table 3 (Part III), a moderate value for each hyper-parameter works best on all the subsets. Too many tools cause confusion, while too few tools cannot train the agent to handle complex tasks. Too many interaction turns cause unreliability for both the agent and the environment backbone due to limited long-horizon capability, yet too few turns make the environment too simple for the agent to learn to handle multi-turn interactions. Overly large step size δ makes the curriculum evolve too rapidly for the agent to establish a solid foundation in easier scenarios, and overly small δ fails to provide sufficient training for complex scenarios.

4.5 Cost Analysis

Table 4 presents the wall time consumed by training. TRUSTEE is within the same order of magnitude as ToolRL and Tool-N1 in training time and much shorter than EnvScaler under the same GPU budget. Note that the 4 GPUs for TRUSTEE's environment backbone are only active during the sampling phase. During optimization, they are idle, so there is no hidden extra budget for TRUSTEE; if anything, the agent itself is trained on fewer GPUs than the baselines that use all 8 GPUs for policy optimization.

Table 4: Training time (hours) on three LMs. The subscript is the relative time to TRUSTEE.

Method	8B-BASE	4B	8B
ToolRL	$110_{2.6\times}$	$31_{1.0\times}$	$21_{0.7\times}$
Tool-N1	$20_{0.5\times}$	$27_{0.9\times}$	$20_{0.7\times}$
EnvScaler	$150_{3.5\times}$	$140_{4.5\times}$	$170_{5.7\times}$
TRUSTEE	$43_{1.0\times}$	$31_{1.0\times}$	$30_{1.0\times}$

Meanwhile, we only compare the training time here. If we count other costs, like data annotation or environment synthesis, other methods will be even more expensive (both in time and resource), while TRUSTEE needs none of these extra steps. Therefore, TRUSTEE is cost-efficient overall.

5 Related Work

5.1 Tool Learning

Tool calling, also known as function calling, allows LMs to obtain information from the external environment [1, 5], make use of helpful utilities [4, 2], and interact with the real world [3], which converts generative assistants to intelligent agents.

Tool learning aims to enhance the tool calling capability of LMs via training. Earlier research focuses on SFT, which requires high-quality offline data. ToolAlpaca [5] generates user queries and tool responses using LMs to synthesize single-turn data. BUTTON [14] and ToolACE [15] simulate both the user and tools to collect multi-turn trajectories. AgentScaler [13] instantiates tools as executable code and local databases and simulates the user with a user agent to collect multi-turn SFT data.

Search-R1 [10] first introduces agentic RL that allows LMs to learn to call tools (*e.g.*, search engine APIs) via multi-turn reinforcement learning, which is superior in generalization. Many works perform RLVR with existing SFT data [7, 8, 9]. To enable truly multi-turn online learning, recent work constructs environments with LM-simulated tools [20] or synthesized code and databases [36, 22, 23]. Our work focuses on scaling the RL environment for tool learning with open-source LMs.

5.2 Environment Scaling for LM Agents

Real-world APIs are usually expensive and unstable [12] while customized tool calling infrastructures require a lot of expert labor, limiting the scalability of tool environments. Therefore, it is necessary to scale the environment for tool calling agents via simulation or synthesis. ZeroSearch [25] replaces the search engine API in Search-R1 with an LM. Simia-RL [20] simulates tool responses in the online interactions with LMs. GTM [37] trains a specific LM to simulate tool responses. AutoForge [36], EnvScaler [22] and AWM [23] automatically generate user intents and build environments in the form of executable code and local databases for online reinforcement learning. Our work instantiates the whole environment as an LM and takes advantage of adaptive curriculum learning.

The novelty of our method lies in: (1) an end-to-end fully simulated environment for tool learning including task generation, user and tool simulation, and LM-based judgment; (2) an adaptive curriculum learning mechanism ensuring smooth training.

6 Conclusion

We have presented TRUSTEE, a cost-friendly method that simulates all the environment components for online tool learning. Empirical results show that TRUSTEE brings improvements in most cases and outperforms most of the compared baselines, which rely on either annotated data or environments synthesized by advanced proprietary LMs. Our work proves that, with a proper design, fully simulated environments can effectively train tool calling agents even with a local open-source 8B LM as the backbone while requiring no annotated data or tasks. Our work moves a step towards democratized tool learning and we hope it can inspire future research on environment scaling with limited resources.

Future work includes: (1) exploring other approaches to fine-grained credit assignment to take advantage of the turn-level rewards in our simulated environment and improve the training with denser supervision; (2) incorporating advanced entropy-controlling techniques to mitigate the entropy decrease during training; (3) incorporating a tool generation module that automatically generates various tools to free TRUSTEE from the reliance on a tool repository.

7 Discussion

Limitations While we provide a strong baseline for tool learning with an 8B LM, our work has certain limitations. First, the scalability of the simulated environment is still constrained due to the bounded capability of the simulation LM, which may not generate sufficiently complex tasks or sufficiently accurate rewards when the task involves a large number of tools and a lot of turns of interactions. Second, there are a lot of hyper-parameters introduced in our method (*e.g.*, those in the adaptive curriculum learning), which makes it difficult to find an optimal setup, especially when the system is sensitive to hyper-parameters. Third, our experiments only validate TRUSTEE within the QWEN3 family, and do not employ other model families like LLAMA [38] due to infrastructure-side compatibility issues caused by different tool-call templates and parsers.

Broader Impact This work democratizes tool learning with a methodology that simulates environments with open-source LMs. By reducing the reliance on human-annotated data, costly real-world infrastructures of executable environments, or expensive proprietary LMs, our work lowers the barrier for researchers to effectively and efficiently train tool calling agents. TRUSTEE can also be easily integrated into hybrid tool learning scenarios and may play the role of warm-up (*e.g.*, using TRUSTEE for the initial training steps) or complement (*e.g.*, using TRUSTEE for unavailable or costly tools). However, there may exist some malicious uses like modifying the prompts and tools to train agents with illegal tools (*e.g.*, hacking, disinformation, harassment). We advocate for the intended uses of TRUSTEE and recommend deploying the trained agents in sandbox environments.

Acknowledgments and Disclosure of Funding

This work is partially supported by the Fundamental Research Funds for the Central Universities, Peking University.

References

[1] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. ToolQA: A dataset for LLM question answering with external tools. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[2] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[3] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[4] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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

[6] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024.

[7] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

[9] Yirong Zeng, Xiao Ding, Yutai Hou, Yuxian Wang, Li Du, Juyi Dai, Qiuyang Ding, Duyu Tang, Dandan Tu, Weiwen Liu, Bing Qin, and Ting Liu. Tool zero: Training tool-augmented LLMs via pure RL from scratch. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

[10] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

[11] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations*, 2024.

[12] Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

[13] Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, Shibin Wu, Zhengwei Tao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Towards general agentic intelligence via environment scaling. *arXiv preprint arXiv:2509.13311*, 2025.

[14] Mingyang Chen, Haoze Sun, Tianpeng Li, Fan Yang, Hao Liang, Keer Lu, Bin Cui, Wentao Zhang, Zenan Zhou, and Weipeng Chen. Facilitating multi-turn function calling for LLMs via compositional instruction tuning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[15] Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Wang Xinzhi, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. ToolACE: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations*, 2025.

[16] Michael Sullivan, Mareike Hartmann, and Alexander Koller. Procedural environment generation for tool-use agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

[17] Tommaso Castellani, Naimeng Ye, Daksh Mittal, Thomson Yen, and Hongseok Namkoong. Synthtools: A framework for scaling synthetic tools for agent development. *arXiv preprint arXiv:2511.09572*, 2025.

[18] Yixia Li, Hongru WANG, Jiahao Qiu, Zhenfei Yin, Dongdong Zhang, Cheng Qian, Zeping Li, Xiaoteng Ma, Guanhua Chen, Heng Ji, and Mengdi Wang. From word to world: Can large language models be implicit text-based world models? In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*, 2026.

[19] Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. Userrl: Training interactive user-centric agent via reinforcement learning. *arXiv preprint arXiv:2509.19736*, 2025.

[20] Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, Lukas Wutschitz, Janardhan Kulkarni, Radha Poovendran, Robert Sim, and Saravan Rajmohan. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025.

[21] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025.

[22] Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis. *arXiv preprint arXiv:2601.05808*, 2026.

[23] Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *arXiv preprint arXiv:2602.10090*, 2026.

[24] Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhong-Zhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026. Survey Certification.

[25] Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*, 2025.

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

[27] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

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

[29] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[30] Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Zhiwei Liu, Yihao Feng, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. xLAM: A family of large action models to empower AI agent systems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.

[31] Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, Jun Wang, and Weinan Zhang. Robust function-calling for on-device language model via function masking. In *The Thirteenth International Conference on Learning Representations*, 2025.

[32] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025.

[33] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[34] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

[35] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

[36] Shihao Cai, Runnan Fang, Jialong Wu, Baixuan Li, Xinyu Wang, Yong Jiang, Liangcai Su, Liwen Zhang, Wenbiao Yin, Zhen Zhang, Fuli Feng, Pengjun Xie, and Xiaobin Wang. Autoforge: Automated environment synthesis for agentic reinforcement learning. *arXiv preprint arXiv:2512.22857*, 2025.

[37] Zhenzhen Ren, Xinpeng Zhang, Zhenxing Qian, Yan Gao, Yu Shi, Shuxin Zheng, and Jiyan He. GTM: Simulating the world of tools for ai agents. *arXiv preprint arXiv:2512.04535*, 2025.

[38] Meta. Llama 3.2. `https://developer.meta.com/ai/docs/model-cards-and-prompt-formats/llama3_2/`, 2026. Accessed: 2026-10-04.

[39] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

A Algorithm of TRUSTEE

We present the algorithm described in pseudo code of TRUSTEE in Algorithm 1. The soft curriculum mechanism is omitted for simplicity.

Algorithm 1 TRUSTEE

Require: Agent policy π_θ; environment backbone LM Φ; tool repository $\mathcal{F}$; number of trajectories per task M; batch size N; training iterations T; initial difficulty $\mathcal{D}_1$; step size δ; reward thresholds $\eta_{\text{high}}, \eta_{\text{low}}$; maximum interaction turns T_{turn}; maximum sequence length L_{seq}

Ensure: Updated policy π_θ

for $n = 1$ **to** T **do**

Stage 1: Task Generation

  $\mathcal{B} \leftarrow \emptyset$ ▷ task batch

  for $i = 1$ **to** N **do**

    $f_i \leftarrow \zeta(\mathcal{F}, \mathcal{D}_n)$; $\Omega_G \leftarrow \Omega(f_i, \mathcal{D}_n)$ ▷ sample tools; build task prompt

    $(p_i, g_i, q_i, c_i) \leftarrow e_0 \sim \mathcal{P}_\Phi(\cdot \mid \Omega_G)$ ▷ generate task

    $h_i \leftarrow \textsc{SystemPrompt}(\mathcal{D}_n)$ ▷ system prompt per difficulty

    $\mathcal{B} \leftarrow \mathcal{B} \cup \{s_0^{(i)} = (f_i, p_i, g_i, q_i, c_i, h_i, \mathcal{D}_n)\}$ ▷ initial state

  end for

Stage 2: Simulated Interaction

  $\mathcal{T} \leftarrow \emptyset$; $\mathcal{R} \leftarrow \emptyset$ ▷ trajectories; final rewards

  for each $s_0^{(i)} \in \mathcal{B}$ **do**

    for $m = 1$ **to** M **do**

      $\tau \leftarrow (h_i, f_i, q_i)$; $t \leftarrow 0$ ▷ initial observation o_0

      repeat

        $a_t \sim \pi_\theta(\cdot \mid \tau)$ ▷ agent action

        $\tau \leftarrow \tau \oplus a_t$ ▷ append action

        if $a_t \in \mathcal{A}_{\text{tool}}$ **then**

          $e_t \sim \mathcal{P}_\Phi(\cdot \mid \tau, \Omega_T), \Omega_T = \Omega(f_i, g_i)$ ▷ tool simulator

        else

          $e_t \sim \mathcal{P}_\Phi(\cdot \mid \tau, \Omega_U), \Omega_U = \Omega(f_i, g_i, p_i)$ ▷ user simulator

        end if

        $\tau \leftarrow \tau \oplus e_t$; $t \leftarrow t + 1$ ▷ append feedback

      until $e_t = \varnothing$ **or** $t \geq T_{\text{turn}}$ **or** $|\tau| \geq L_{\text{seq}}$ ▷ terminate

      $r_{i,m} \sim \mathcal{R}_\Phi(\cdot \mid \tau, \Omega_V), \Omega_V = \Omega(f_i, g_i, p_i, c_i, \mathcal{D}_n)$ ▷ verify trajectory

      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{r_{i,m}\}$ ▷ collect

    end for

  end for

Stage 3: Policy Optimization

  $\hat{A} \leftarrow \textsc{GroupRelativeAdvantage}(\mathcal{R})$ ▷ advantage from final reward

  $\pi_\theta \leftarrow \text{GRPO}\textsc{Update}(\pi_\theta, \mathcal{T}, \hat{A})$ ▷ policy optimization

Difficulty Evolution

  $\bar{r}_n \leftarrow \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} r$ ▷ average reward of the batch

  if $\bar{r}_n > \eta_{\text{high}}$ **then**

    $\mathcal{D}_{n+1} \leftarrow \mathcal{D}_n + \delta$ ▷ too easy

  else if $\bar{r}_n < \eta_{\text{low}}$ **then**

    $\mathcal{D}_{n+1} \leftarrow \mathcal{D}_n - \delta$ ▷ too hard

  else

    $\mathcal{D}_{n+1} \leftarrow \mathcal{D}_n$ ▷ unchanged

  end if

end for

return π_θ

B Prompt Templates

The prompt templates for task generation, tool simulation, user simulation, and verifier simulation are presented in Figure 3, 4, 5, and 6, respectively.

C Theoretical Justifications

Below we provide more detailed justifications for Proposition 2.1. For simplicity, we assume $\Phi = \theta_0$, which means the environment backbone shares the same foundation model with the initial agent

Prompt Template for Task Generation

```
You are an expert at creating realistic tasks for agent training. Given a set of tools and difficulty settings, generate ONE task.

## Available Tools
{tools}

## Settings
- Expected number of distinct tool calls the agent should make: approximately {number of expected calls}
- User persona (user's expertise; describe this in the task): {persona}
- Query ambiguity (for the first user message only): {ambiguity}. User persona and query ambiguity are independent (e.g. an expert can ask vaguely; a novice can ask clearly).
- Expected interaction turns: {number of turns}

## Your Task
Produce a single task with:
1. Expected turns: The task as a whole should match expected turns: {number of turns} Design the goal and first message so the conversation length fits this.
2. expected tool calls: A list of tool names (from the available tools above) that the agent is expected to use, in a plausible order. Length approximately {number of expected calls}.
3. user intent: A clear description of what the user ultimately wants to achieve (the ground truth goal for evaluation). This may be richer or more specific than what the user says in the first message.
4. user persona: A short description of the user's expertise level, matching: {persona}. Do NOT tie this to query clarity.
5. first user query: The user's first message only—how a real user would actually open the conversation. Keep it natural:
   - It need not state the full intent directly; the user might hint, ask a partial question, or express a vague need that the agent must clarify.
   - Match the query ambiguity level: {ambiguity}
   - Do not name tools; the message should lead naturally toward using the expected tools as the conversation unfolds.
   - Avoid sounding like a spec or a list of requirements; write as a real person would (e.g. "I'm trying to figure out..." or "Can you help with something?" rather than "I want you to do X, Y, and Z").

## Output Format (JSON only)
{
   "expected_tool_calls": ["tool_name_1", "tool_name_2", ...],
   "user_intent": "Description of what the user wants to achieve.",
   "user_persona": "Short description of user expertise (not query clarity).",
   "first_user_query": "The user's first message (ambiguity as specified above)."
}

Generate the task now. Output only valid JSON.
```

Figure 3: Prompt template for task generation.

policy, to ablate the effect of standard distillation (*i.e.*, $\text{Gain}_{\text{trans}} \approx 0$). Then, we justify that $\Delta J > 0$ by demonstrating the environment provides a training signal with higher information density than unguided autonomous self-play by the agent itself.

Distillation via Privileged Context. The learning process can be formalized as the maximization of mutual information $I(\tau; z)$ between the agent's trajectory τ and the hidden knowledge $z = \{g, p, c, \mathcal{D}_n\}$ related to the ground truth. While the agent is restricted to $\mathcal{I}_{\text{agent}}$, the environment transition $\mathcal{P}_\Phi(e_t|\tau, \Omega_{T/U})$ is a function of the privileged information. Therefore, each observation e_t acts as a signal in a noisy channel conveying knowledge regarding the ground truth. By optimizing the policy against the privileged verifier's reward conditioned on z, the agent minimizes the conditional entropy $H(z|\tau)$, distilling the privileged knowledge of the environment into a reactive policy. This ensures that the policy converges to the manifold of trajectories that are mutually informative of the ground truth goal even without direct access to z.

The Generator-Verifier Complexity Gap. The computational asymmetry between generation and verification justifies the validity of the reward signal from the environment. Let $\pi_\theta(\tau)$ be the generative distribution and $V(\tau, z)$ be the verification function (the value of which is in $\{-1, 0, 1\}$ in our implementation). To solve the task, the agent has to search a massive action space to obtain a successful trajectory. On the contrary, the task for the verifier is deterministic: assessing whether the trajectory τ satisfies z. Consequently, the entropy of the environment's verification $H(V(\tau, z)|\mathcal{I}_{\text{env}})$ is strictly lower than that of the generative distribution of the agent $H(\pi_\theta(\tau)|\mathcal{I}_{\text{agent}})$ (*i.e.*, $H_{\text{ver}} \ll H_{\text{gen}}$). Therefore, the verifier possesses a higher signal-to-noise ratio and acts as a filter truncating the agent's exploration distribution in the RL process. Even when $\Phi = \theta_0$, the lower complexity of the discriminative task ensures that the verifier provides a more informative gradient than an agent would provide through unguided self-play.

Prompt Template for Tool Simulation

You are an environment simulator for tool calling agents. Your task is to simulate realistic tool execution results based on the interaction history and the agent's tool call.

Interaction History
{history}

Agent's Tool Call
{tool calling}

Available Tools
{tools}

User Intent (task context)
{user intent}

Your Task
Simulate the execution of the agent's tool call and return:
1. Tool Result: The simulated execution output (or an error message if the call is invalid)
2. Tool Reward: An integer reward in {-1, 0, 1} evaluating how well this tool call serves the user's intent

Tool Result Guidelines
Validation
First, validate the tool call:
- Parseable: Is the tool call properly formatted and parseable? If the tool call is null or malformed, return an error like "Error: Tool call must be properly formatted in the expected structure"
- Tool exists: Is the tool name in the available tools?
- Required arguments: Are all required parameters provided?
- Argument types: Do argument types match specifications?
- Argument values: Are values realistic and contextually appropriate?
If invalid, return an error message explaining the issue.
Simulation (for valid calls)
If valid, simulate realistic execution output:
- Contextual: Use information from history to generate relevant results
- Realistic: Output should mirror real tool behavior with concrete, specific data
- Consistent: Maintain consistency with previous interactions
- Plausible: Avoid placeholders or generic values; use realistic naming
Examples of good simulation:
- Instead of "file1.txt", use "ProjectReport 2024 Q1.pdf"
- Instead of "user@example.com", use "sarah.chen@techcorp.com"
- Instead of "ID: 123", use "ID: a7c3f91b-2e8d"
Examples of error messages:
- "Error: Tool call is malformed or could not be parsed"
- "Error: Tool 'send email' not found in available tools"
- "Error: Missing required parameter 'recipient' for tool 'send email'"
- "Error: Parameter 'amount' must be a number, got string"

Tool Reward Guidelines
Assign exactly one of -1, 0, or 1:
1 — Tool call fully serves the user's intent
  - Valid and well-formed (correct tool, required arguments present and correctly typed)
  - Clearly advances the user's goal; optimal or near-optimal choice at this step
  - Arguments are accurate and contextually appropriate
0 — Tool call is valid but suboptimal
  - Correctly formatted and executable (tool exists, arguments valid)
  - Only partly aligned with the user's intent, or a reasonable but not best choice
  - May be redundant, slightly off-purpose, or use weaker arguments than possible
-1 — Invalid or intent-mismatched
  - Invalid: malformed, unparseable, wrong tool, missing or wrong-typed arguments, or will fail
  - Or valid but not addressing the user's intent: wrong task, irrelevant tool, or counterproductive for the goal
First decide validity and intent alignment, then assign the single value {-1, 0, 1}. Malformed or unparseable tool calls are always -1.

Output Format
Return your response in the following JSON format:
```
{
  "result": "The simulated tool execution output or error message",
  "reward": 0
}
```

Important:
- `result` must be a string (success output or error message; may be JSON inside the string if needed)
- `reward` must be exactly one of: -1, 0, 1
- Output valid JSON only (no extra text or comments)

Generate the simulation now.

Figure 4: Prompt template for tool simulation.

Prompt Template for User Simulation

You are simulating a human user interacting with an assistant agent to accomplish a task.

Original Task / First User Message
{user query}

User Intent (what the user ultimately wants)
{user intent}

User Persona
{user persona}

Available Tools (for agent use)
{tools}

Interaction History
{history}

Agent's Latest Message
{agent message}

Your Task
As the user, respond to the agent's latest message and evaluate its quality.
Response Guidelines
Be Human-Like
- Use natural, conversational language
- Be concise and to the point (typically 1-3 sentences)
- Show realistic reactions and emotions when appropriate
- Don't repeat the exact task; use your own words
Provide Information Appropriately
- Only provide information when the agent asks for it
- Don't volunteer all details at once
Know When to End
- If the task is successfully completed, respond with an empty string ""
- If the agent has clearly failed and cannot help, respond with an empty string ""
- Only end when it's genuinely appropriate to stop the conversation
Be Realistic
- React naturally to agent's actions (confusion, satisfaction, frustration, anger, etc.)
- Ask for clarification if the agent's response is unclear
- Point out errors or issues if the agent makes mistakes
Reward Guidelines
Assign exactly one of -1, 0, or 1 based on how well the agent's message serves the user's intent and moves the task forward:
1 — Significant progress; fully meets the user's intent
  - Clearly addresses what the user asked or needed at this step
  - Moves the task meaningfully toward completion (e.g. correct information, right tool use, useful next step)
  - Response is on-topic, clear, and appropriate
0 — Little progress or suboptimal
  - Partly relevant or only marginally helpful
  - Some progress but slow, indirect, or with unnecessary detours
  - Suboptimal choices (e.g. weaker tool use or incomplete answer) but not wrong
-1 — Wrong, unclear, no progress, or off-topic
  - Wrong answer, off-topic, or fails to address the user's intent
  - Unclear or confusing so the user cannot act on it
  - No real progress (e.g. repetition, irrelevant actions, or counterproductive steps)
First judge progress and intent alignment, then assign the single value {-1, 0, 1}.

Output Format
Return your response in the following JSON format:
{
"response": "Your message as the user",
"reward": 0
}

Important:
- `response` must be a string; use an empty string "" if the conversation should end
- `reward` must be exactly one of: -1, 0, 1
- Output valid JSON only (no extra text or comments)

Generate the user simulation now.

Figure 5: Prompt template for user simulation.

Prompt Template for Verifier Simulation

```
Your task is to evaluate how well an assistant agent performed in helping a user accomplish their task.

## User's Original Query
{user query}

## Expected Tool Calls (should be used correctly)
{expected tool calls}

## User Intent (ground truth)
{user intent}

## User Persona
{user persona}

## Available Tools
{tools text}

## Complete Interaction Trajectory
{interaction history text}

## Evaluation Task
Evaluate the assistant agent's overall performance using ONLY the following {num criteria} key criteria (in order of importance):
{criteria text}
### Reward Scale
Assign a final reward of 1, 0, or -1:
- 1: All of the above {num criteria} key criteria are met.
- 0: Most of the above {num criteria} key criteria are met.
- -1: Most of the above {num criteria} key criteria are NOT met or the agent does not call the tools at all.

## Output Format
Return your evaluation in the following JSON format:
{
"reward": 0,
"reasoning": "Brief explanation (1-2 sentences)"
}

Important:
- reward must be exactly one of: -1, 0, 1
- Output valid JSON only

Generate the evaluation now.
```

Figure 6: Prompt template for verifier simulation.

Manifold Expansion and Latent Unfolding. Finally, we address the state-space coverage. Let $\mathcal{M}_{\text{pre}}$ be the manifold of knowledge acquired by the LM during pre-training. The zero-shot agent π_θ typically collapses to a high-probability sub-manifold $\mathcal{M}_{\text{active}} \subset \mathcal{M}_{\text{pre}}$. Therefore, the transition dynamics induced by the agent LM $\mathcal{P}_\theta$ is limited to high-probability scenarios. Nevertheless, the structural prompts Ω in the environment act as force-fields that shift the transition dynamics $\mathcal{P}_\Phi$, which expands the reachable state-space:

$$\text{Vol}\left(\text{supp}(\mathcal{P}_\Phi|\Omega)\right) > \text{Vol}\left(\text{supp}(\mathcal{P}_\theta)\right), \tag{7}$$

where Vol refers to the volume of the state-space manifold and supp denotes the support (the set of reachable scenarios with reasonably large probability). Due to this expansion, the agent is forced to interact with long-tail scenarios (*e.g.*, a novice user or a highly ambiguous query), through which it unfolds latent capabilities that come from pre-training but are dormant in its default policy. Therefore, the environment can be seen to serve as a mechanism of alignment that projects the broad latent knowledge into a manifest policy.

In summary, the above three mechanisms facilitate a bootstrap effect collectively. By virtue of the simulated environment which is more informative and structurally diverse than the default reachable space of the agent, TRUSTEE enables an LM to transcend its initial performance limits. This theoretical framework demonstrates how an LM can act as its own supervisor, unfolding its latent potential into a specialized policy. The results of QWEN3-8B in our experiments also validate the structural gain empirically.

Detailed System Prompt

You are a helpful multi-turn dialogue assistant capable of leveraging tool calls to solve user tasks.

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
If you need to call tools, use the following format:
```
<think> Your thoughts and reasoning </think>
<tool_call> Tool call JSON object(s) </tool_call>
```
If you need to respond to the user, use the following format:
```
<think> Your thoughts and reasoning </think>
Your response to the user.
```

Important Notes
1. You must always include the `<think>` field to outline your reasoning. Provide either calling tools or responding to the user. The former leads to an interaction with the tool environment, while the latter switches back to the user interface.
2. You may invoke multiple tool calls simultaneously in the `<tool_call>` field. You will get the tool feedback after calling the tools. You may call tools for multiple turns if needed.
3. Refer to the previous dialogue records in the history, including the user's queries, previous tool calls, tool feedback, and your responses.

Figure 7: Detailed system prompt.

D Details of Optimization

We sample M trajectories (each including T_m interaction steps with $|a_t|$ tokens each) for each task $\mathcal{T}_i$ and apply multi-turn GRPO [10] with token-level loss [39]:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathop{\mathbb{E}}_{\mathcal{T}_i \sim \mathcal{T}} \left[\frac{1}{Z} \sum_{m=1}^{M} \sum_{t=0}^{T_m - 1} \sum_{j=1}^{|a_t|} \Big(\text{CLIP}(r_{m,t,j}(\theta), \hat{A}_{m,t,j}, \varepsilon) - \beta \, \mathrm{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \Big) \right], \tag{8}$$

where Z is the normalization factor:

$$Z = \sum_{m=1}^{M} \sum_{t=0}^{T_m - 1} |a_t|, \tag{9}$$

$r_{m,t,j}(\theta)$ is the importance sampling ratio:

$$r_{m,t,j}(\theta) = \frac{\pi_\theta\big(x_{m,t,j} \mid \tau_{m,<t},\, x_{<j}\big)}{\pi_{\theta_{\text{old}}}\big(x_{m,t,j} \mid \tau_{m,<t},\, x_{<j}\big)}, \tag{10}$$

$\hat{A}_{m,t,j}$ is the group-relative advantage based on the final reward only:

$$\hat{A}_{m,t,j} = \frac{r_m \;-\; \text{Mean}\big(\{r_m\}_{m=1}^{M}\big)}{\text{Std}\big(\{r_m\}_{m=1}^{M}\big)}, \tag{11}$$

and $\mathrm{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$ is the KL penalty term.

E More Implementation Details

E.1 Details of Task Difficulty

The detailed descriptions of the different difficulty dimensions at all levels are presented in Table 5. Note that the rewarding criteria are applied in an incremental manner. The criteria at level l should include all those from level 0 to level l.

E.2 Details of Tool Repository

We preprocess the raw tool repository with QWEN3-NEXT-80B-A3B-INSTRUCT[5], discarding low-quality tools. The prompt template for the LM is presented in Figure 9.

[5]https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct

Table 5: Details of task difficulty.

Aspect	Level	Description
Interaction Turns	0	One or a couple of turns: the task should be resolvable in a short exchange (e.g. 1–2 back-and-forths).
	1	A few to several turns: the task should naturally involve some back-and-forth (e.g. clarification, multiple steps, a handful of exchanges).
	2	Many turns: the task should require an extended conversation (e.g. multi-step workflow, several clarifications, or many tool uses and replies).
System Prompt	0	See Figure 7.
	1	See Figure 8.
	2	You are a helpful multi-turn dialogue assistant capable of leveraging tool calls to solve user tasks.
	3	You are a helpful assistant.
User Persona	0	Expert: the user has extensive prior knowledge in the domain.
	1	Beginner: the user has some prior knowledge but may need guidance.
	2	Novice: the user has no prior knowledge and may need clarification on basics.
Query Ambiguity	0	Clear: the first user message should be specific, complete, and unambiguous.
	1	Somewhat ambiguous: the first user message may omit some details or be a little vague.
	2	Highly ambiguous: the first user message should be vague, incomplete, or missing key information that the agent may need to clarify.
Rewarding Criteria	0	Meet the user's intent: Did the agent fulfill what the user wanted by using the tools?
	1	Correctly use the expected tools: Did the agent use the expected tools appropriately and correctly?
	2	Free of hallucination: No invented facts, tool names, or outputs.
	3	No redundant or incorrect trials: No unnecessary or wrong tool calls or steps.
	4	Concise and efficient: Response and tool use are to-the-point and efficient.

Concise System Prompt

You are a helpful multi-turn dialogue assistant capable of leveraging tool calls to solve user tasks.

Important Notes
1. You must always include the `<think>` field to outline your reasoning. Provide either calling tools or responding to the user. The former leads to an interaction with the tool environment, while the latter switches back to the user interface.
2. You may invoke multiple tool calls simultaneously in the `<tool_call>` field. You will get the tool feedback after calling the tools. You may call tools for multiple turns if needed.
3. Refer to the previous dialogue records in the history, including the user's queries, previous tool calls, tool feedback, and your responses.

Figure 8: Concise system prompt.

Prompt Template for Tool Preprocessing

You are a tool quality evaluator. Determine if this tool should be kept based on:
Evaluation Criteria
1. Tool Name: Must be meaningful (not generic like "api", "hello", "test") 2. Description: Must be informative (not just repeating the name or single letters) 3. Parameters: Must have reasonable descriptions (not just "Parameter: {name}") 4. Overall Quality: Tool should be useful and clear
Tool to Evaluate
{tool}
Response Format
Respond ONLY with "KEEP" or "DISCARD" followed by a brief reason.
Format: KEEP/DISCARD: reason

Figure 9: Prompt template for tool preprocessing.

E.3 Details of Evaluation

For BFCL V4, we follow the default evaluation setup and do not further extend the maximum context size of the LM for long-context scenarios. Please note that our evaluated BFCL V4 scores may not be directly comparable to the official leaderboard[6], because we are using a newer version of the evolving BFCL V4 (committed in March 2026) and inference backends may differ (we are using local `vLLM` while the leaderboard does not specify this).

For τ^2-bench, we adopt GPT-4.1 as the user simulator following [27]. The maximum number of interaction turns is 64 in our experiments due to limited budget. The cost for each run is about $30.

E.4 Details of Baselines

ToolRL We reproduce ToolRL based on their released scripts[7] (we do not directly run theirs due to model incompatibility) with aligned maximum sequence length. We ran the training for 500 steps with a batch size of 32 to align the overall training time. However, the training for QWEN3-8B collapsed soon after 400 steps. Therefore, we evaluate the checkpoint of 400 steps for QWEN3-8B.

Tool-N1 We reproduce Tool-N1 based on their released scripts[8] (we do not directly run theirs due to model incompatibility) with aligned maximum sequence length. We ran the training for 500 steps with a batch size of 32 to align the overall training time.

AWM We evaluate their released checkpoint[9] trained using QWEN3-4B and QWEN3-8B directly because no training script is released and we cannot reproduce the method. There is no available checkpoint for QWEN3-8B-BASE, thus there is no result of AWM for this LM.

EnvScaler We train the LM with the provided scripts[10] and the default hyper-parameters. We focus on the pure RL setting and thus train the model from scratch without SFT, which is in line with TRUSTEE and other baselines.

Simia-RL We evaluate their released checkpoint[11] trained using QWEN3-8B on τ-bench directly because re-running the training was infeasible within budget due to Simia-RL's reliance on expensive proprietary LMs. Meanwhile, there is no available checkpoint for QWEN3-8B-BASE or QWEN3-4B, thus there is no result of Simia-RL for these LMs.

E.5 Hyper-parameters

The hyper-parameters for training are listed in Table 6. The batch size is only 16 because a larger batch size requires a higher level of concurrency which may cause the local `vLLM` server for the simulation LM to be overloaded.

[6] `https://gorilla.cs.berkeley.edu/leaderboard.html`
[7] `https://github.com/qiancheng0/ToolRL`
[8] `https://github.com/NVlabs/Tool-N1`
[9] `https://huggingface.co/Snowflake/Arctic-AWM-8B`
[10] `https://github.com/RUC-NLPIR/EnvScaler`
[11] `https://huggingface.co/Simia-Agent/Simia-Tau-SFT-RL-Qwen3-8B`

Table 6: Hyper-parameters for training.

Hyper-parameter	Value
Learning Rate	1e-6
Total Steps	200
Batch Size (Number of Tasks per Step)	16
Mini Batch Size	4
Micro Batch Size	1
KL Loss Coefficient	0.001
Clip Ratio	0.2
Temperature	1.0
Group Size	8
Maximum Prompt Length	8,192
Maximum Response Length	16,384

E.6 Details for Hyper-parameter Ablations

Table 7: Setting for different levels of hyper-parameters.

Setting	Tool	User Turn	Tool Turn	Agent Turn	δ
Default	10	3	3	6	3
Tool↑	20	3	3	6	3
Tool↓	5	3	3	6	3
Turn↑	10	5	5	10	3
Turn↓	10	1	1	2	3
Step↑	10	3	3	6	10
Step↓	10	3	3	6	1

The detailed values of different hyper-parameters used in Section 4.4 are presented in Table 7.

F Extended Experimental Results

F.1 Confidence Intervals of the Main Results

Table 8: Main results with confidence intervals.

Method	BFCL (Non-Live)	BFCL (Live)	BFCL (Multi-Turn)	τ^2-bench (Airline)	τ^2-bench (Retail)	τ^2-bench (Telecom)
QWEN3-8B-BASE	8.38 [6.82, 10.03]	4.74 [3.66, 6.08]	0.00 [0.00, 0.48]	26.00 [15.9, 39.6]	4.39 [1.9, 9.9]	1.75 [0.5, 6.2]
ToolRL	40.83 [38.17, 43.66]	56.33 [53.67, 58.94]	0.25 [0.00, 0.97]	30.00 [19.1, 43.8]	6.14 [3.0, 12.1]	5.26 [2.4, 11.0]
Tool-N1	37.85 [34.85, 40.81]	29.53 [27.16, 32.02]	0.12 [0.02, 0.70]	32.00 [20.8, 45.8]	4.39 [1.9, 9.9]	4.39 [1.9, 9.9]
EnvScaler	75.29 [72.46, 78.06]	62.03 [59.41, 64.58]	0.62 [0.27, 1.45]	28.00 [17.5, 41.7]	6.14 [3.0, 12.1]	0.00 [0.0, 3.3]
TRUSTEE	83.92 [81.44, 86.25]	72.54 [70.10, 74.85]	7.88 [6.20, 9.95]	32.00 [20.8, 45.8]	5.26 [2.4, 11.0]	6.14 [3.0, 12.1]
QWEN3-4B	87.77 [85.56, 89.85]	82.16 [80.03, 84.11]	32.12 [28.98, 35.44]	22.00 [12.8, 35.2]	27.19 [19.9, 36.0]	9.65 [5.5, 16.5]
ToolRL	79.44 [76.75, 82.08]	71.65 [69.19, 73.99]	13.00 [10.85, 15.51]	2.00 [0.4, 10.5]	11.40 [6.8, 18.5]	0.00 [0.0, 3.3]
Tool-N1	88.44 [86.38, 90.42]	77.28 [74.97, 79.43]	34.12 [30.92, 37.48]	24.00 [14.3, 37.4]	9.65 [5.5, 16.5]	11.40 [6.8, 18.5]
AWM	87.12 [84.96, 89.23]	80.61 [78.41, 82.63]	30.00 [26.93, 33.26]	20.00 [11.2, 33.0]	22.81 [16.1, 31.3]	9.65 [5.5, 16.5]
EnvScaler	86.50 [84.23, 88.67]	82.24 [80.11, 84.18]	38.75 [35.44, 42.17]	28.00 [17.5, 41.7]	29.82 [22.2, 38.8]	10.53 [6.1, 17.5]
TRUSTEE	88.96 [86.90, 90.96]	81.35 [79.18, 83.33]	35.62 [32.38, 39.00]	22.00 [12.8, 35.2]	28.07 [20.6, 36.9]	16.67 [10.9, 24.6]
QWEN3-8B	87.77 [85.65, 89.77]	80.31 [78.11, 82.34]	39.88 [36.54, 43.31]	24.00 [14.3, 37.4]	42.98 [34.3, 52.2]	15.79 [10.2, 23.6]
ToolRL	88.77 [86.67, 90.75]	81.20 [79.03, 83.19]	41.25 [37.89, 44.70]	22.00 [12.8, 35.2]	16.67 [10.9, 24.6]	14.04 [8.8, 21.6]
Tool-N1	87.83 [85.71, 89.88]	78.90 [76.65, 81.00]	44.38 [40.97, 47.84]	28.00 [17.5, 41.7]	36.84 [28.6, 46.0]	10.53 [6.1, 17.5]
AWM	87.52 [85.35, 89.62]	80.46 [78.26, 82.49]	39.75 [36.42, 43.18]	30.00 [19.1, 43.8]	36.84 [28.6, 46.0]	12.28 [7.5, 19.6]
EnvScaler	87.42 [85.25, 89.54]	81.20 [79.03, 83.19]	44.50 [41.09, 47.96]	22.00 [12.8, 35.2]	4.39 [1.9, 9.9]	9.65 [5.5, 16.5]
TRUSTEE	89.65 [87.67, 91.56]	82.16 [80.03, 84.11]	46.38 [42.94, 49.84]	26.00 [15.9, 39.6]	45.61 [36.8, 54.8]	17.54 [11.7, 25.6]

We report the main results with 95% confidence intervals in Table 8. For QWEN3-8B-BASE on BFCL, TRUSTEE's intervals do not overlap with the original model's, supporting its significant gain. Yet there are certain overlaps of intervals when the policy LM has already been post-trained (QWEN3-4B and QWEN3-8B), for both TRUSTEE and other compared methods. From a relative perspective, TRUSTEE remains the strongest or tied strongest method by point estimate in 12 out of 18 model-benchmark settings in the table, while using no annotated data and no proprietary LM.

F.2 Effect of Turn-level Fine-grained Reward

We explore integrating the turn-level fine-grained reward given by the backbone LM simulating the user or tool executor. The advantage for the t-th turn is given by:

$$\hat{A}_t = \begin{cases} \frac{r_t+1}{2} & \text{if } r_t \geq 0 \\ \frac{r_t-1}{2} & \text{if } r_t < 0 \end{cases}, \tag{12}$$

where r_t is the reward of the t-th turn. Therefore, a positive turn-level reward leads to a positive advantage, while a negative reward leads to a negative advantage. Meanwhile, a reward with a higher absolute value leads to a larger-magnitude advantage.

Table 9: Analysis on the advantage.

Advantage	Non-Live	Live	Multi-Turn
Sequence-level	**88.96**	81.35	**35.62**
Turn-level	88.31	**81.94**	33.00

We replace the default sequence-level advantage with this turn-level one and experiment with QWEN3-4B. As shown in Table 9, the fine-grained advantage does not necessarily provide further improvements and even exhibits a degradation on the Multi-Turn subset. This indicates that the default sequence-level advantage already provides reliable performance. Therefore, we adopt the sequence-level advantage by default in our experiments.

Nevertheless, this is a naïve design of turn-level advantages, which does not guarantee the zero-mean property and may not support high-quality training. We leave a more advanced turn-level advantage calculation for future work.

F.3 Effect of the Environment Backbone LM

Table 10: Analysis on the environment backbone LM.

Agent LM θ	Environment LM Φ	Non-Live	Live	Multi-Turn
QWEN3-8B-BASE	QWEN3-8B	**83.92**	72.54	**7.88**
	QWEN3.5-397B-A17B-FP8	78.42	**73.58**	2.38
QWEN3-4B	QWEN3-8B	**88.96**	**81.35**	35.62
	QWEN3.5-397B-A17B-FP8	87.23	80.01	**37.88**
QWEN3-8B	QWEN3-8B	**89.65**	**82.16**	**46.38**
	QWEN3.5-397B-A17B-FP8	88.35	79.20	41.25

We replace the environment backbone with QWEN3.5-397B-A17B-FP8[12], which is a more advanced LM (but quantized), to analyze the effect of the backbone LM. Due to the large size of this LM, we use 8 GPUs for serving the environment backbone and another 8 GPUs for training the agent.

As shown in Table 10, the effect of the backbone LM is complex. The more advanced backbone LM exhibits improvements in the Live subset for QWEN3-8B-BASE and in the Multi-Turn subset for QWEN3-4B, but fails to bring further gains in other cases. This may be due to the downgraded performance of the FP8-quantized LM and certain limitations in sparse models. Anyway, we can still conclude that QWEN3-8B is already a good choice for the backbone of TRUSTEE, which brings improvements with high efficiency.

Due to a limited budget, we cannot afford to further experiment with other advanced backbones, especially proprietary LMs. In our experiments, each complete training requires an order of magnitude of 10^8 tokens from the backbone LM approximately, which demands an extremely high cost.

F.4 Simulating Stateful Tools

Like the simulation quality evaluation in Section 4.2, we use the stateful tools from EnvScaler [22] as the tool repository and sample 100 complete trajectories via QWEN3-8B self-play in the simulated environment for DEEPSEEK-V4-PRO to judge the simulation quality across all aspects (from 1 to 5). The overall simulation quality is also healthy, with all aspects scoring above 4.0. To be specific, the scores for task generation, user simulation, tool simulation, final reward, internal consistency, and overall fidelity are 4.65, 4.37, 4.82, 4.20, 4.72, and 4.36, respectively.

[12] https://huggingface.co/Qwen/Qwen3.5-397B-A17B-FP8

To further study the simulation performance of stateful tools, we also train QWEN3-8B-BASE with the stateful tools used above under our main experimental settings and report the end-to-end results on BFCL in Table 11. The stateful setting still brings significant improvement to the base model. Nevertheless, its gains are smaller than the default stateless setting, especially on the multi-turn subset, which may reflect both the harder stateful-tool setup and the narrower tool and task coverage due to a much smaller tool repository.

Table 11: Performance of stateful tool simulation on BFCL with QWEN3-8B-BASE.

Setting	Non-Live	Live	Multi-Turn
Qwen3-8B-Base	8.38	4.74	0.00
TRUSTEE (default)	**83.92**	**72.54**	**7.88**
TRUSTEE (stateful)	74.60	51.15	1.00

G Computational Resources

All our experiments are conducted on a machine with TencentOS Server 4, 384 AMD® EPYC™ 9K84 96-Core Processor CPUs and 2.2TiB memory. We use 8× NVIDIA H20 GPUs for both training and evaluation for all the experiments (except for the experiments in Appendix F.3).

H Use of AI Assistants

The algorithmic design and core methodology of this work were derived through manual research and human reasoning. We use AI assistants solely for wording, editing, formatting, and coding purposes. They do not impact the core methodology, scientific rigor, or originality of the research.

I Use of Scientific Artifacts

We cite all the creators of scientific artifacts we use in this paper. Licenses and URLs of these scientific artifacts are shown in Table 12. Our use of these is consistent with their intended use.

Table 12: URLs and licenses of scientific artifacts we use.

Artifact	URL	License
Qwen3 Models	`https://huggingface.co/collections/Qwen/qwen3`	Apache-2.0
Qwen3.5 Models	`https://huggingface.co/collections/Qwen/qwen35`	Apache-2.0
verl	`https://github.com/volcengine/verl`	Apache-2.0
vLLM	`https://github.com/vllm-project/vllm`	Apache-2.0
BFCL	`https://github.com/ShishirPatil/gorilla`	Apache-2.0
τ^2-bench	`https://github.com/sierra-research/tau2-bench`	MIT
ToolBench	`https://github.com/openbmb/toolbench`	Apache-2.0
ToolRL	`https://github.com/qiancheng0/ToolRL`	Apache-2.0
Tool-N1	`https://github.com/NVlabs/Tool-N1`	Apache-2.0
EnvScaler	`https://github.com/RUC-NLPIR/EnvScaler`	MIT
AWM	`https://huggingface.co/collections/Snowflake/agent-world-model`	Apache-2.0
Simia-RL	`https://huggingface.co/Simia-Agent/Simia-Tau-SFT-RL-Qwen3-8B`	MIT

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contributions and scope of this paper are accurately reflected by the main claims made in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Discussed in Section 7.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not include theoretical results (*i.e.*, theorems and proofs), but only provides theoretical analysis with conceptual propositions and justifications (Section 2.4 and Appendix C).

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Details for reproduction are provided in Section 3, Appendix B and E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided the URL to our code and data in Section 1.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Provided in Section 3, Appendix B and E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: It is highly computationally expensive to train and evaluate the models. We cannot afford to run multiple times to report statistical significance. Yet we provide the confidence intervals of our main results in Appendix F.1, which is not run-to-run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Provided in Section 4.5 and Appendix G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper conforms with the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Discussed in Section 7.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: To the best of our knowledge, this paper poses no major risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Discussed in Appendix I.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Discussed in Appendix H.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.