Tools as Continuous Flow for Evolving Agentic Reasoning

Tairan Huang[1,*], **Siyu Shang**[2,*], **Qiang Chen**[1,*,†], **Xiu Su**[2,‡], **Yi Chen**[1,‡]
[1]The Hong Kong University of Science and Technology
[2]Central South University
tairanhuang99@gmail.com, siyushang@csu.edu.cn,
qchencw@connect.ust.hk, xiusu1994@csu.edu.cn, yichen@ust.hk

Abstract

Large Language Models (LLMs) have demonstrated remarkable capabilities in orchestrating external tools for complex reasoning tasks. However, existing methods predominantly rely on a step-wise paradigm that optimizes one discrete decision at a time, limiting their global planning perspective and causing errors to accumulate over long horizons. To overcome these limitations, we propose Tools as Continuous Flow for Evolving Agentic Reasoning (FlowAgent), which reconceptualizes discrete tool chaining as continuous trajectory generation within a semantic space. At each environment phase, a single conditional-flow solve generates the complete provisional future tool trajectory, with every latent anchor corresponding to exactly one tool at that plan position. We further construct a plan-level supervision and evaluation protocol from public tool-use resources by pairing each observed execution prefix with its complete remaining trajectory under strict forward-only execution. Theoretically, we establish formal bounds on utility convergence and prove that our continuous formulation fundamentally guarantees robust generalization and error attenuation. Empirical evaluations show that FlowAgent achieves superior robustness and adaptability in long-horizon reasoning tasks. Our code and datasets are available at https://github.com/Tairan-Terrian/FlowAgent.

1 Introduction

Large Language Models (LLMs) have evolved into capable agents that utilize external tools [1, 2]. By interacting with search engines, APIs, and transactional systems, these models ground their reasoning in real-world environments [3–6]. This paradigm allows agents to execute complex operations and substantially expands their utility beyond static text generation [7–9]. Consequently, they are increasingly used to automate sophisticated workflows requiring sequential logic and dynamic environment interaction [10–13].

Recent methods formulate tool planning as an autoregressive generation task over a discrete action space [14–17]. At each step, the agent observes the current context, evaluates the available options, and selects one tool for execution. Reinforcement learning algorithms [18–20] and prompting strategies [21–23] can improve these local decisions, but the underlying process remains rooted in next-action generation and does not explicitly model the structure of the complete remaining plan.

These step-wise paradigms exhibit three fundamental limitations when handling complex scenarios, as illustrated in Figure 1. (1) *Lack of global perspective.* Existing methods optimize localized tool selection but do not directly represent how the current decision participates in the remaining

[*]Equal Contribution.
[†]Leading the Project.
[‡]Corresponding Authors.

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

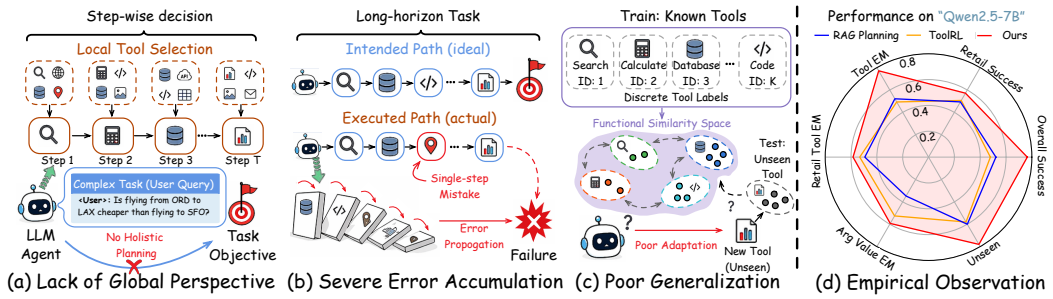


Figure 1: (a-c) Conceptual illustrations of three limitations observed in discrete tool-reasoning paradigms: *limited global perspective*, *error accumulation*, and *weak generalization to unseen tools*. As evidenced in (d), the performance of the traditional step-wise paradigm is inadequate under the plan-level benchmark, highlighting the need for a paradigm shift toward global plan-level reasoning.

workflow. (2) *Severe error accumulation.* In long-horizon tasks, a single suboptimal choice can alter the environment and derail the subsequent reasoning chain. (3) *Poor generalization to unseen tools.* Treating tools as isolated categorical labels obscures their functional similarities and weakens transfer to novel schemas.

To address these inherent limitations, we propose Tools as Continuous Flow for Evolving Agentic Reasoning (FlowAgent), a novel paradigm that fundamentally shifts from myopic step-wise selection to global plan-level reasoning by reconceptualizing discrete tool chaining as continuous trajectory generation within a semantic space. To systematically evaluate this paradigm, we introduce the first closed-loop benchmark dedicated to plan-level agentic reasoning in dynamic real-world environments. Specifically, the proposed FlowAgent formulates multi-step planning as continuous trajectory evolution supervised by expert demonstrations, providing a global planning perspective to ensure coherent reasoning. These latent plans are subsequently translated into precise discrete actions within a closed-loop execution scheme, which dynamically absorbs environmental feedback and explicitly aligns the entire process with optimal plan-level utility. Theoretically, we establish formal bounds on utility convergence and mathematically prove that our continuous formulation guarantees semantic-driven tool generalization and fundamental error attenuation., encoder choices, trajectory targets, replanning horizons, and inference overhead.

The contributions of this work are summarized below.

- **Plan-level Benchmark:** We introduce the plan-level closed-loop benchmark dedicated to plan-level agentic reasoning in dynamic real-world environments. By integrating diverse real-world domains, this setup systematically evaluates the ability to dynamically absorb environmental feedback during multi-step execution.
- **Novel Paradigm:** We propose FlowAgent, a novel paradigm for evolving agentic reasoning. FlowAgent reconceptualizes discrete tool chaining as continuous trajectory generation within a semantic space via conditional flow matching, which shifts the decision process from myopic step-wise selection to global plan-level reasoning.
- **Theoretical Foundation:** We provide a rigorous mathematical framework that establishes explicit bounds on plan utility convergence and generalization to unseen tools. Furthermore, we formally prove that our closed-loop execution scheme strictly attenuates error accumulation in long-horizon reasoning tasks.
- **Empirical Superiority:** Extensive experiments demonstrate that FlowAgent significantly outperforms baseline methods, exhibiting superior robustness in long-horizon tasks and adaptability to unseen toolsets.

2 Related Work

LLM-based Tool Reasoning. The efficacy of LLM agents relies heavily on their ability to interact with external tools [24, 25]. Early paradigms attempted to mitigate this through explicit prompt

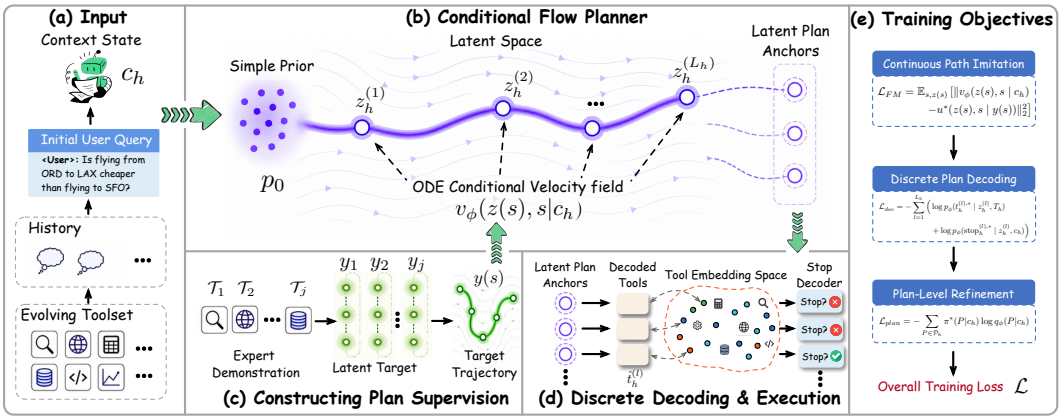


Figure 2: The overall framework of **FlowAgent**, which reconceptualizes discrete tool chaining as continuous trajectory generation to shift from myopic step-wise selection to plan-level reasoning, including *Conditional Flow Planner*, *Constructing Plan Supervision*, *Discrete Decoding and Execution*, and *Training Objectives*.

engineering [26–28] or interleaving chain-of-thought reasoning with API invocations [29–31]. Recent advancements have transitioned towards autonomous tool integration [32–35], formalizing the selection and execution of diverse APIs. ToolRL [14], ToolPlanner [36], Agent-R1 [19], and AutoTIR [33] optimize discrete tool interactions through trajectory rewards, path feedback, or modular agentic RL. However, these systems predominantly operate within a discrete action space, treating tool planning as isolated categorical selections rather than globally coherent trajectories. In contrast, FlowAgent reconceptualizes discrete tool chaining as continuous trajectory generation within a semantic space, shifting the decision process from myopic step-wise selection to global plan-level reasoning to unlock robust adaptability to unseen toolsets.

LLMs Agent Planning. LLMs Agent planning is essential for decomposing complex goals into executable sub-tasks, while simple autoregressive generation often fails to maintain long-term coherence [37–40]. Early methods focused on linear reasoning chains [41, 42] or iterative self-correction [43, 44] to refine outputs based on internal heuristics. Recent research has moved toward sophisticated search-based architectures to explore and evaluate multiple reasoning branches [45, 46]. Furthermore, specialized planners [47, 48] incorporate environmental feedback to adjust planning trajectories during execution. These methods still rely on discrete search within a step-wise framework and can accumulate errors over long horizons. In contrast, FlowAgent bypasses the limitations of myopic discrete search by generating continuous trajectories, facilitating global plan-level reasoning that maintains structural coherence.

3 Methodology

3.1 Problem Definition and Formalization

We first introduce problem definition and formalization, followed by a detailed description of the design of FlowAgent. The overall framework is shown in Figure 2.

Environment, State, and Tool Representation. Consider an agentic reasoning task evolving over discrete phases $h \in \{1, \dots, H\}$. At phase h, the system maintains an available toolset $T_h = \{t_1, \dots, t_{|T_h|}\}$, which dynamically evolves over time. To enable unseen tool generalization, each tool $t \in T_h$ is parameterized by a continuous semantic embedding $e_t \in \mathbb{R}^d$, jointly encoding its functional description and usage constraints. We define the context state at phase h as $c_h = (x, \tau_{<h}, T_h)$, where x denotes the initial query, and $\tau_{<h} = \big((r_1, \hat{t}_1, o_1), \dots, (r_{h-1}, \hat{t}_{h-1}, o_{h-1})\big)$ encapsulates the historical reasoning trajectory. Specifically, at each past step $i < h$, r_i denotes the generated rationale, $\hat{t}_i$ is the executed tool, and o_i represents the environmental observation returned after execution.

We distinguish the *environment phase* h from the *future plan position* l. At one phase h, FlowAgent generates all L_h future positions jointly. It does not invoke the planner L_h times.

Instead of directly predicting discrete token-level actions, our framework formally models a continuous latent plan object:

$$P_h = (z_h^{(1)}, z_h^{(2)}, \ldots, z_h^{(L_h)}), \tag{1}$$

where each latent anchor $z_h^{(l)} \in \mathbb{R}^d$ corresponds to a future planning step. Through discrete decoding, this continuous path projects to a provisional tool sequence $\hat{t}_h^{(l)} \in T_h$ for $l \in \{1, \ldots, L_h\}$, with exactly one latent anchor assigned to each plan position.

Plan-Level Objective. Traditional step-wise tool selection focuses on optimizing a localized policy $t_h \sim \pi(t|c_h, T_h)$. In contrast, we elevate the objective to model a plan-level continuous distribution $P_h \sim q_\phi(P|c_h)$. The execution decision relies on an embedding-based decoder mapping the continuous anchor to candidate tools via a distance-based probability:

$$p_\phi(t|z_h^{(l)}, T_h) \propto \exp\left(-\frac{||z_h^{(l)} - e_t||_2^2}{\epsilon}\right), \tag{2}$$

where ϵ is the temperature scaling factor controlling the discrete decoding margin.

To systematically address the global structure of multi-tool tasks, the overall training target is formulated as plan utility maximization $\max_\phi \mathbb{E}[U(P_h; c_h)]$. The utility function U explicitly unifies task performance with practical constraints:

$$U(P_h; c_h) = \text{Acc}(P_h; c_h) - \lambda_{cost}\text{Cost}(P_h) - \lambda_{red}\text{Red}(P_h) + \lambda_{cons}\text{Cons}(P_h; c_h), \tag{3}$$

where $\text{Acc}(\cdot)$ measures final reasoning quality, $\text{Cost}(\cdot)$ penalizes redundant calls or latency, $\text{Red}(\cdot)$ penalizes cyclic failures, $\text{Cons}(\cdot)$ measures alignment with the observation context, and $\lambda_{\{cost,red,cons\}} > 0$ are balancing hyperparameters.

Closed-Loop Planning Objective. Since tool outputs inevitably alter subsequent environmental configurations, we explicitly formalize a closed-loop execution paradigm. Upon receiving the observation o_h from the executed tool $\hat{t}_h$, the context state is deterministically updated via the operator $\mathcal{U}$:

$$c_{h+1} = \mathcal{U}(c_h, \hat{t}_h, o_h). \tag{4}$$

Unlike open-loop planners that rigidly execute long-horizon chains, our system dynamically resamples the latent plan conditioned on the updated state:

$$P_{h+1} \sim q_\phi(P|c_{h+1}). \tag{5}$$

Consequently, the approach operates as a receding-horizon latent plan generator, using real feedback to limit the propagation of stale early-stage predictions.

3.2 Conditional Flow Planner

We define $s \in [0, 1]$ as continuous planning time within one environment phase. Given c_h, the latent path $z_h(s)$ is governed by an ordinary differential equation (ODE) driven by a conditional velocity field v_ϕ:

$$\frac{dz_h(s)}{ds} = v_\phi(z_h(s), s|c_h), \quad z_h(0) \sim p_0, \tag{6}$$

where p_0 is a simple prior distribution. Let $s_l = (l-1)/(L_h - 1)$ for $L_h > 1$ (and $s_1 = 1$ for $L_h = 1$). A single numerical solve retains the path states at these positions:

$$P_h = (z_h^{(1)}, z_h^{(2)}, \ldots, z_h^{(L_h)}), \qquad z_h^{(l)} := z_h(s_l). \tag{7}$$

Each anchor $z_h^{(l)} \in \mathbb{R}^d$ serves as a continuous semantic directive for exactly one future plan position. Crucially, the anchors in P_h are obtained together from one conditional-flow solve rather than from repeated next-tool predictions.

3.3 Constructing Plan Supervision

Training the conditional flow planner requires continuous target trajectories derived from discrete expert demonstrations. Given an expert reasoning chain of length m, formally denoted as $\mathcal{T}^* =$

$\big((r_1, t_1, o_1), \ldots, (r_m, t_m, o_m)\big)$, we construct a dense latent plan state $y_j \in \mathbb{R}^d$ for each discrete step $j \in \{1, \ldots, m\}$:

$$y_j = W_t e_{t_j} + W_r \text{Enc}(r_j) + W_o \text{Enc}(o_{j-1}) + W_p \nu_j, \tag{8}$$

where o_{j-1} denotes the previous observation (a learned null-observation token is used for $j = 1$), e_{t_j} represents the semantic tool embedding, ν_j is a learnable phase embedding, and $W_{\{t,r,o,p\}}$ are projection matrices. By default, Enc is the frozen transformer trunk of Qwen2.5-7B-Instruct used only as a text feature extractor.

For each phase h of the expert trace, the observed prefix $(x, T_h, \tau_{<h})$ forms the conditioning state c_h, while the complete suffix $(y_h, \ldots, y_m)$ forms the trajectory target. Thus, a length-m trace yields m distinct prefix-to-remaining-trajectory pairs, and no future action is exposed in c_h. For a suffix of length $L_h = m - h + 1$, relabel its anchors as $\tilde{y}_l = y_{h+l-1}$ for $l = 1, \ldots, L_h$. To formulate the regression objective over normalized time $s \in [0, 1]$, we construct a continuous target path $y_h(s)$ by piecewise linear interpolation at $s_l = (l-1)/(L_h - 1)$:

$$y_h(s) = \tilde{y}_l + \frac{s - s_l}{s_{l+1} - s_l}(\tilde{y}_{l+1} - \tilde{y}_l), \quad s \in [s_l, s_{l+1}]. \tag{9}$$

For $L_h = 1$, the path is constant. This trajectory is the explicit regression target for the conditional velocity field.

3.4 Discrete Decoding and Execution

To instantiate the continuous latent plan into executable actions, the framework maps the generated latent anchor $z_h^{(l)}$ to the dynamically evolving discrete tool space T_h. According to the distance-based distribution defined above, every latent anchor is first decoded into a provisional tool:

$$\hat{t}_h^{(l)} \sim p_\phi(t | z_h^{(l)}, T_h). \tag{10}$$

To infer the remaining plan length and prevent over-generation, we concurrently learn a parameterized stop mechanism. This module estimates the termination probability for each plan position conditioned on the local anchor and the global context state:

$$p_\phi(\text{stop} | z_h^{(l)}, c_h). \tag{11}$$

The first position whose probability exceeds a confidence threshold marks the provisional end of the decoded trajectory.

Crucially, we eschew the rigid open-loop execution of the entire planned sequence P_h. At phase h, the agent executes strictly the leading tool $\hat{t}_h^{(1)}$ corresponding to the first latent anchor. The system retrieves the real-world observation o_h and deterministically updates the context state:

$$c_{h+1} = \mathcal{U}(c_h, \hat{t}_h^{(1)}, o_h). \tag{12}$$

Following this localized execution, the remaining unexecuted anchors $(z_h^{(2)}, \ldots, z_h^{(L_h)})$ are discarded. A new continuous latent trajectory P_{h+1} is dynamically resampled conditioned on the updated state c_{h+1}, establishing a formally defined plan-execute-replan loop.

3.5 Training Objectives

To optimize the conditional flow planner and discrete execution, we propose a three-stage training pipeline, including *Continuous Path Imitation*, *Discrete Plan Decoding*, and *Plan-Level Refinement*.

Continuous Path Imitation. We first train the flow planner to imitate the expert trajectories via conditional flow matching. The predicted velocity field v_ϕ is optimized to match the teacher vector field u^* induced by the interpolated target path $y(s)$:

$$\mathcal{L}_{FM} = \mathbb{E}_{s, z(s)} \left[||v_\phi(z(s), s | c_h) - u^*(z(s), s | y(s))||_2^2 \right]. \tag{13}$$

Discrete Plan Decoding. We optimize the embedding-based tool decoder and the parameterized stop mechanism. Given the ground-truth tool identity $t_h^{(l),*}$ and the position-specific stopping indicator $\text{stop}_h^{(l),*} \in \{0, 1\}$, we minimize the aggregated cross-entropy loss over the extracted anchors:

$$\mathcal{L}_{dec} = -\sum_{l=1}^{L_h} \Big(\log p_\phi(t_h^{(l),*} | z_h^{(l)}, T_h) + \log p_\phi(\text{stop}_h^{(l),*} | z_h^{(l)}, c_h) \Big). \tag{14}$$

Plan-Level Refinement. Finally, to align the generated latent trajectories with the global multi-tool objective, we perform plan-level reinforcement learning over sampled candidate plans $P \in \mathcal{P}_h$. We bridge the utility ranking with policy optimization:

$$\mathcal{L}_{plan} = -\sum_{P \in \mathcal{P}_h} \pi^*(P|c_h) \log q_\phi(P|c_h), \tag{15}$$

where the target policy distribution $\pi^*(P|c_h)$ is weighted by holistic plan utility function $U(P; c_h)$:

$$\pi^*(P|c_h) \propto q_{\text{old}}(P|c_h) \exp(U(P; c_h)), \tag{16}$$

and $q_{\text{old}}(P|c_h)$ represents the generative policy from the previous optimization iteration.

The overall training target is formulated as a weighted combination of these objectives:

$$\mathcal{L} = \mathcal{L}_{FM} + \lambda_1 \mathcal{L}_{dec} + \lambda_2 \mathcal{L}_{plan} + \lambda_3 \mathcal{L}_{cons}, \tag{17}$$

where $\lambda_{\{1,2,3\}} > 0$ are balancing coefficients, and $\mathcal{L}_{cons} = \frac{1}{L_h} \sum_{l=1}^{L_h} ||z_h^{(l)} - W_c \text{Enc}(c_h)||_2^2$ acts as an auxiliary constraint explicitly regularizing the semantic consistency between the generated latent anchors and the prevailing observational context.

4 Theoretical Analysis

We next connect continuous path approximation to discrete decoding, utility deviation, unseen-tool transfer, and feedback-conditioned error propagation. These are finite-horizon sufficient-condition results, not universal guarantees: each conclusion applies only when the corresponding geometric, margin, regularity, and feedback assumptions below hold.

4.1 Foundational Assumptions

To ensure the tractability of the latent planning problem, we introduce the following structural assumptions based on the system's geometric properties.

Assumption 1 (Semantic Regularity)**.** *The train and test tool embeddings lie in a compact region of a smooth manifold $\mathcal{M} \subset \mathbb{R}^d$. On this region, the excess planning risk induced by replacing one tool embedding with another is $C_{\mathcal{M}}$-Lipschitz in their Euclidean distance.*

Assumption 2 (Decoding Margin)**.** *For every expert anchor y_j with ground-truth tool t^*, there exists $\Delta_{\text{dec}} > 0$ such that $\|y_j - e_{t^*}\|_2 + \Delta_{\text{dec}} \leq \|y_j - e_t\|_2$ for all $t \in T_h \setminus \{t^*\}$.*

Assumption 3 (Utility Regularity)**.** *For the finite horizons considered, U is bounded and Lipschitz in both the latent plan and encoded context: $|U(P; c) - U(P'; c')| \leq L_P \sum_l \|z^{(l)} - z'^{(l)}\|_2 + L_c d_c(c, c')$.*

Assumption 4 (Feedback Stability)**.** *Let $e_h = d_c(c_h, c_h^*)$ be the encoded-context deviation from an oracle execution and let $a_h \geq 0$ collect the new flow, decoding, and observation error introduced at phase h. The feedback-conditioned update satisfies $e_{h+1} \leq \rho e_h + a_h$ for some local $0 \leq \rho < 1$ on the evaluated trajectory region.*

4.2 Propositions and Theorems

Proposition 1 (Endpoint Error Bound)**.** *Let the teacher field be K_u-Lipschitz and let the pointwise velocity error be at most δ. Then $\eta_\phi := \mathbb{E}\|z(1) - y(1)\|_2 \leq C_{\text{flow}} \delta$, where $C_{\text{flow}} = (e^{K_u} - 1)/K_u$ (or 1 when $K_u = 0$). For nearest-neighbor/MAP decoding,*

$$\Pr(\hat{t}_{\text{MAP}} \neq t^*) \leq \min\left\{1, \frac{2\eta_\phi}{\Delta_{\text{dec}}}\right\}. \tag{18}$$

Remark 1. *If the endpoint error is deterministically below $\Delta_{\text{dec}}/2$, the correct tool remains the unique nearest neighbor; Eq.* (18) *covers the expected-error case. The detailed proof is provided in Appendix A.1.*

Theorem 1 (Plan Utility Convergence)**.** *Let P_h^* be the expert plan and let P_h be the generated plan of length L_h. Suppose the anchor neighborhood has positive squared-distance gaps $g_l > 0$,*

the maximum one-tool utility change is B_U, and $N_h = |T_h|$. If η_h bounds the expected integrated velocity error, then

$$|U(P_h^*; c_h) - \mathbb{E}U(P_h; \hat{c}_h)| \le L_P L_h C_{\text{flow}} \eta_h + B_U(N_h - 1) \sum_{l=1}^{L_h} e^{-g_l/\epsilon} + L_c \beta_o \mathbb{E}\|o - \hat{o}\|_2. \tag{19}$$

Consequently, the expected utility approaches the oracle value as the flow and observation errors vanish and the decoder becomes deterministic while retaining positive gaps.

Remark 2. *The three terms isolate continuous path approximation, finite-temperature decoding, and feedback perturbation; the theorem does not assert convergence when a margin collapses or the stability assumptions fail. The detailed proof is provided in Appendix A.2.*

Theorem 2 (Unseen Tool Generalization)**.** *For every unseen tool, let $\pi(t)$ be its nearest training proxy. Under Assumption 1,*

$$\mathbb{E}[\text{Err}_{\text{unseen}}] \le \mathbb{E}[\text{Err}_{\text{proxy}}] + C_{\mathcal{M}}(\delta_{\text{shift}} + \epsilon_{\text{cover}}), \tag{20}$$

where δ_{shift} is the distance from the unseen region to the training manifold region and ϵ_{cover} is the covering radius of the training embeddings.

Remark 3. *The bound separates the proxy's existing planning error from the additional penalty caused by semantic shift and incomplete training coverage. The detailed proof is in Appendix A.3.*

Proposition 2 (Closed-loop Error Contraction)**.** *Under Assumption 4, the cumulative closed-loop context error obeys*

$$E_{\text{closed}} \le \sum_{i=1}^{H-1} a_i \frac{1 - \rho^{H-i}}{1 - \rho} \le \sum_{i=1}^{H-1} (H - i) a_i =: E_{\text{open}}. \tag{21}$$

The second inequality is strict whenever some $a_i > 0$ has at least two subsequent phases and $\rho < 1$.

Remark 4. *Here E_{open} is the no-correction accumulation envelope for the same local errors. The result characterizes replanning only inside the assumed contractive region; it does not cover non-contractive or adversarial updates. The detailed proof is provided in Appendix A.4.*

5 Experiments

5.1 Benchmark and Experimental Setup

Plan-level Supervision and Evaluation Protocol. We build on four public sources: τ^2*-bench* [49], *ToolBench* [50], *API-Bank* [51], and *GTA* [52]. Our contribution is not a new closed-loop environment: τ^2-bench supplies the executable environment. Instead, we standardize schemas and traces and create a plan-level protocol in which every observed expert prefix is paired with its complete remaining trajectory. The unified data contain 6,865 tasks, 3,930 tool schemas, and 599 test-only tools. For stateful executable evaluation, each prediction is replayed against real tools, policies, and databases under strict forward-only execution; the submitted split contains 140 trajectories and the expanded study uses up to 800. More detailed descriptions of the dataset construction and execution configurations are provided in Appendix B.1.

Baselines. We compare FlowAgent with Direct-LLM, Standard SFT, RAG Planning, and ToolRL [14] in the main table. Expanded comparisons add ToolPlanner [36], Agent-R1 [19], and AutoTIR [33] under matched executors and budgets; implementation details and full results appear in Appendix B.2 and Appendix C.

Metrics. To holistically evaluate the global planning capabilities and execution robustness, we employ a multi-dimensional metric suite, including Overall Success, Retail Success, Tool EM, Retail Tool EM, Pred Exec, and Arg Value EM. The detailed formulations for all evaluation metrics are provided in Appendix B.3.

5.2 Experimental Results

Main Results. As shown in Table 1, FlowAgent consistently outperforms all baselines across both executors. Compared to the Standard SFT baseline lacking the continuous flow matching

Table 1: Main results on our proposed plan-level benchmark. All models are evaluated under a strict forward-execution regime. Highlighted are the results ranked **first**, second, and third.

Executor	Methods	Execution Success		Planning Accuracy		Execution & Grounding	
		Overall Success ↑	Retail Success ↑	Tool EM ↑	Retail Tool EM ↑	Pred Exec ↑	Arg Value EM ↑
Qwen2.5-7B -Instruct	Direct-LLM	0.1214 ± 0.0186	0.2208 ± 0.0215	0.1286 ± 0.0179	0.2338 ± 0.0231	0.9412 ± 0.0068	0.4308 ± 0.0197
	Standard SFT	0.6571 ± 0.0126	0.3766 ± 0.0174	0.6714 ± 0.0118	0.4026 ± 0.0192	0.9268 ± 0.0075	0.4387 ± 0.0156
	RAG Planning	0.5214 ± 0.0158	0.4935 ± 0.0189	0.5214 ± 0.0163	0.4935 ± 0.0195	**1.0000** ± 0.0000	0.3478 ± 0.0172
	ToolRL	0.4786 ± 0.0289	0.5065 ± 0.0316	0.4929 ± 0.0274	0.5325 ± 0.0331	0.9643 ± 0.0098	0.5257 ± 0.0295
	FlowAgent (Ours)	**0.7643** ± 0.0098	**0.5714** ± 0.0137	**0.7745** ± 0.0091	**0.5892** ± 0.0152	0.9619 ± 0.0061	**0.5929** ± 0.0124
Llama-3.2-3B -Instruct	Direct-LLM	0.2000 ± 0.0207	0.0779 ± 0.0169	0.2071 ± 0.0198	0.0909 ± 0.0182	0.8060 ± 0.0116	0.1225 ± 0.0175
	Standard SFT	0.6214 ± 0.0152	0.3117 ± 0.0201	0.6286 ± 0.0144	0.3247 ± 0.0213	0.9318 ± 0.0084	0.4348 ± 0.0187
	RAG Planning	0.5286 ± 0.0176	0.5065 ± 0.0198	0.5500 ± 0.0169	0.5455 ± 0.0211	0.9091 ± 0.0102	0.3241 ± 0.0181
	ToolRL	0.1357 ± 0.0361	0.1169 ± 0.0328	0.1500 ± 0.0348	0.1429 ± 0.0355	0.9362 ± 0.0124	0.1028 ± 0.0307
	FlowAgent (Ours)	**0.7571** ± 0.0112	**0.5584** ± 0.0159	**0.7714** ± 0.0103	**0.5844** ± 0.0167	**0.9524** ± 0.0072	**0.5573** ± 0.0136

Table 2: Ablation studies of proposed components on the Qwen2.5-7B executor. Additional ablation results can be found in Appendix C.

Flow Planner	Con. State	Stop Mech.	Overall Suc.	Retail Suc.	Tool EM	Retail Tool EM	Pred Exec	Arg Value EM
×	×	×	0.6571	0.3766	0.6714	0.4026	0.9268	0.4387
✓	×	×	0.6786	0.4156	0.6857	0.4286	0.9381	0.5099
✓	✓	×	0.7357	0.5195	0.7429	0.5325	**0.9619**	0.5613
✓	✓	✓	**0.7643**	**0.5714**	**0.7745**	**0.5892**	**0.9619**	**0.5929**

prior, our method improves the overall success rate on Qwen2.5-7B from 0.6571 to 0.7643. This performance gap indicates that our generative prior offers better state conditioning than simple retrieval hints or discrete policy optimization. Finally, the consistent performance gains observed during the Llama-3.2-3B evaluation validate FlowAgent as a generalized planning framework capable of bridging abstract reasoning and structured action execution in long-horizon reasoning tasks.

Ablation Studies. To evaluate the contribution of each component, we conduct the ablation study shown in Table 2. Starting from Standard SFT, introducing the Conditional Flow Planner increases Overall Success from 0.6571 to 0.6786 and improves Arg Value EM from 0.4387 to 0.5099. Adding the observation-grounded Context State produces the largest improvement and raises Overall Success to 0.7357. Finally, the Parameterized Stop Mechanism further improves Retail Success from 0.5195 to 0.5714 and Retail Tool EM from 0.5325 to 0.5892. These results demonstrate that each component provide complementary benefits for long-horizon tool orchestration.

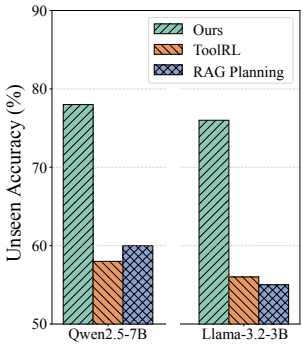


Figure 3: Generalization performance on unseen tools.

Generalization Results. To evaluate generalization to novel tool schemas, we measure planning accuracy on a strictly held-out set of unseen tools, as shown in Figure 3. FlowAgent achieves a planning accuracy of 78% on the Qwen2.5-7B executor and 76% on the Llama-3.2-3B executor. FlowAgent's consistent performance indicates that continuous semantic trajectories capture tool relationships that remain useful when the available toolset changes.

RL Baseline Results. To determine whether the improvement extends beyond ToolRL, we compare FlowAgent with ToolPlanner, Agent-R1, and AutoTIR, as shown in Figure 4. FlowAgent nevertheless improves Overall Success over Agent-R1 from 0.7186 to 0.7643 on Qwen2.5-7B and from 0.6929 to 0.7571 on Llama-3.2-3B. These results confirm that the improvement is consistent across recent RL planner families and is not explained by executability alone.

Larger-Executor Results. To evaluate whether FlowAgent remains effective as executor capacity increases, we repeat the comparison on four larger instruction-tuned models, as shown in Figure 5. ToolRL becomes stronger on larger executors, particularly on the retail subset, but remains below FlowAgent in task success. The consistent improvements within both model families demonstrate that complete-trajectory supervision remains beneficial as the underlying executor scales.

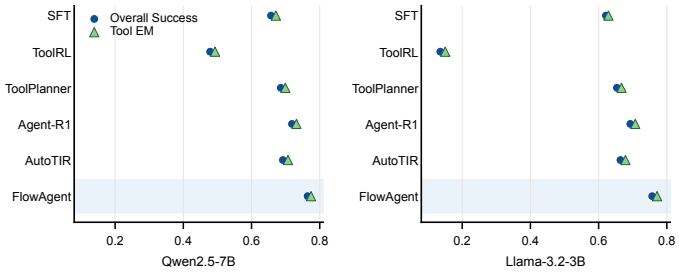


Figure 4: Comparison with recent RL tool-use methods.

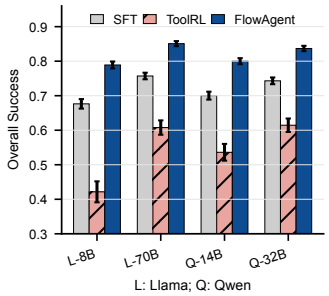


Figure 5: Larger executors.

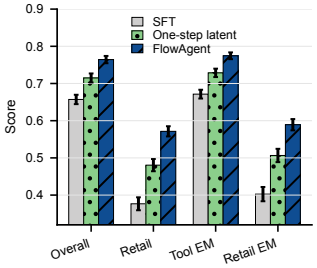


Figure 6: Prediction target.

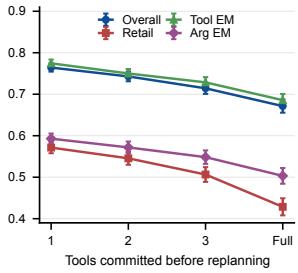


Figure 7: Replanning frequency.

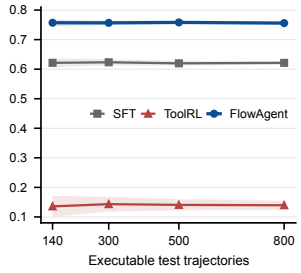


Figure 8: Test-set sizes.

5.3 Additional Analysis and Discussion.

Prediction-Target Analysis. To distinguish complete-trajectory generation from a stronger one-step semantic representation, we compare three prediction targets in Figure 6. One-step latent regression improves Overall Success from 0.6571 to 0.7149 and Retail Success from 0.3766 to 0.4805 over discrete SFT, confirming that semantic regression itself is useful. FlowAgent further improves these metrics to 0.7643 and 0.5714 when the entire future latent trajectory is supervised.

Replanning Analysis. Figure 7 evaluates how many predicted tools should be executed before incorporating new feedback. Retail Success exhibits a larger decline from 0.5714 to 0.4286 because database-changing observations can invalidate later actions in the provisional trajectory. This result demonstrates that complete-trajectory generation supplies global planning structure, while frequent feedback-conditioned replanning remains necessary for reliable execution.

Test-Scale Analysis. To examine whether the submitted 140-trajectory test set produces a favorable estimate, we construct independent test sets containing 300, 500, and 800 executable trajectories. As shown in Figure 8, FlowAgent's Overall Success remains between 0.7559 and 0.7582 across all three test sizes, and Tool EM varies by at most 0.0033. These results show that the original performance advantage is stable under substantially expanded executable evaluation.

Efficiency Analysis. To evaluate computational efficiency, we compare the training overhead of FlowAgent with the reinforcement-learning baseline ToolRL. As reported in Table 3, FlowAgent completes optimization in 12.29 minutes, whereas ToolRL requires 48.75 minutes. The peak memory on the primary GPU is reduced from 78.69 to 52.63 GiB, and the average peak memory across all three GPUs decreases from 68.36 to 55.14 GiB. This improvement results from learning the conditional velocity field through direct trajectory regression, which avoids the repeated environment rollouts and policy-update overhead required by discrete RL optimization. Therefore, FlowAgent improves planning performance while substantially reducing training time and memory consumption.

Table 3: Computational efficiency comparison. We report the total training time and the peak memory consumption across three distinct GPUs.

Method	Training Time ↓	GPU 1 Peak ↓	GPU 2 Peak ↓	GPU 3 Peak ↓	Avg Peak ↓
ToolRL	48.75 min	78.69G	63.22G	63.16G	68.36G
FlowAgent	**12.29 min**	**52.63G**	**59.91G**	**52.88G**	**55.14G**

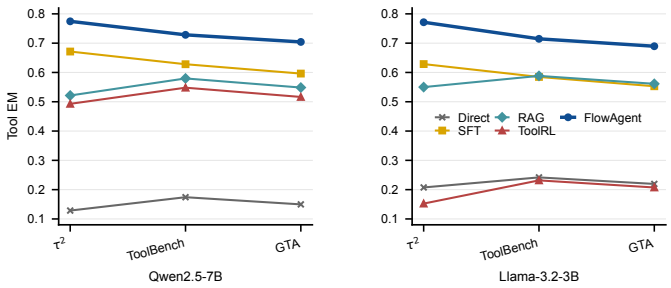


Figure 9: Tool EM across task sources.

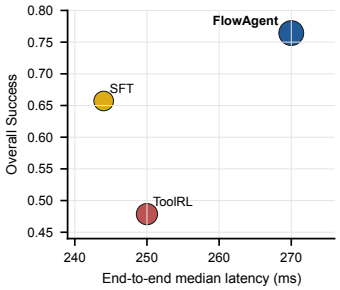


Figure 10: Inference latency versus.

Table 4: Temperature sensitivity.

Temperature	Overall Suc. ↑	Retail Suc. ↑	Tool EM ↑
$\epsilon = 0.01$	0.6957	0.5165	0.7029
$\epsilon = 0.05$	0.7351	0.5452	0.7504
$\epsilon = 0.10$	**0.7643**	**0.5714**	**0.7745**
$\epsilon = 0.20$	0.7214	0.5329	0.7286
$\epsilon = 0.50$	0.6729	0.4975	0.6971
$\epsilon = 0.60$	0.6540	0.4826	0.6815

Table 5: Retail decision phases.

Method	Decision Case	Success ↑	Arg Value EM ↑
FlowAgent	Initial Lookup	0.7000	0.6833
	Detail Lookup	0.7692	0.6923
	Database Operation	**0.3529**	**0.4202**
- w/o Stop Mech.	Initial Lookup	0.7000	0.6833
	Detail Lookup	0.7692	0.6923
	Database Operation	0.1176	0.2878
Oracle Selection	Database Operation	0.7059	0.6190

Cross-Source Analysis. To test whether the improvement is confined to a single benchmark source, we regroup the generated predictions by τ^2-bench, ToolBench, and GTA. Figure 9 shows that FlowAgent obtains the highest Tool EM on every source for both executors. The results demonstrate that FlowAgent's advantage is not produced by one benchmark's tool distribution.

Inference-Cost Analysis. Figure 10 reports the deployment trade-off introduced by numerical flow integration. FlowAgent uses 16 velocity-field function evaluations per generated plan, increasing median planning latency from 171 to 194 ms and median end-to-end latency from 244 to 270 ms relative to SFT.

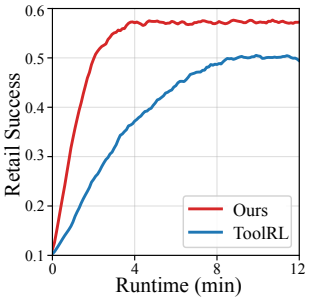


Figure 11: Convergence speed.

Parameter Analysis. To evaluate robustness to the decoder configuration, we vary the temperature ϵ and report the results in Table 4. FlowAgent obtains its best performance at $\epsilon = 0.10$, reaching 0.7643 Overall Success, 0.5714 Retail Success, and 0.7745 Tool EM. The smooth performance variation around the optimum indicates that FlowAgent is not sensitive to a narrowly tuned temperature value.

Retail Decision Analysis. The main evaluation shows that state-changing database operations are the most difficult decisions in the retail subset. We therefore separate the initial lookup, detail lookup, and database-operation phases in Table 5. The oracle tool-selection result reaches 0.7059 success, showing that state-conditioned operation selection and argument grounding remain the primary bottlenecks in complex retail tasks.

Convergence Analysis. To compare optimization behavior, we track Retail Success as a function of training time in Figure 11. FlowAgent improves rapidly during the first few minutes and stabilizes after approximately 4 minutes at a higher Retail Success score. In contrast, ToolRL requires about 9 minutes to approach convergence. This result is consistent with the training-cost comparison and shows that FlowAgent improves both convergence speed and final task performance.

6 Conclusion

In this paper, we present **FlowAgent**, a novel framework that reconceptualizes tools as continuous flow for evolving agentic reasoning. FlowAgent leverages conditional flow matching to generate continuous trajectories and provides a global perspective to ensure coherent reasoning and robust tool execution. Theoretically, we establish formal bounds on utility convergence and prove that our formulation fundamentally guarantees error attenuation. Extensive empirical evaluations demonstrate that FlowAgent achieves superior adaptability in long horizon reasoning tasks.

Acknowledgments and Disclosure of Funding

This research is partially supported by grants from the National Natural Science Foundation of China (#72495125 and #W2511077). Yi Chen is supported by the Hong Kong Research Grants Council, Early Career Scheme Fund [Grant 26508924] and Hong Kong RGC Theme-based Research Scheme T32-615-24/R. This research is also funded by the National Natural Science Foundation of China 62406347.

References

[1] Lu Dai, Ziyang Rao, Yili Wang, Hanqing Wang, Hao Liu, and Hui Xiong. Towards mechanistically understanding why memorized knowledge fails to generalize in large language model finetuning. *arXiv preprint arXiv:2607.08393*, 2026.

[2] Yili Wang, Lu Dai, Tairan Huang, Yijie Xu, and Hui Xiong. Vl-eraser: Vacuum distillation for machine unlearning in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31504–31513, 2026.

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

[4] Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. Llm agents making agent tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26092–26130, 2025.

[5] Xinzhe Li. A review of prominent paradigms for llm-based agents: Tool use (including rag), planning, and feedback learning. *arXiv preprint arXiv:2406.05804*, 2024.

[6] Xiu Su, Qinghua Mao, Zhongze Wu, Xi Lin, Shan You, Yue Liao, and Chang Xu. Large language models driven neural architecture search for universal and lightweight disease diagnosis on histopathology slide images. *NPJ digital medicine*, 8(1):682, 2025.

[7] Weikai Xu, Chengrui Huang, Shen Gao, and Shuo Shang. Llm-based agents for tool learning: A survey: W. xu et al. *Data Science and Engineering*, pages 1–31, 2025.

[8] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small llms are weak tool learners: A multi-llm agent. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 16658–16680, 2024.

[9] Zhongze Wu, Hongyan Xu, Yitian Long, Shan You, Xiu Su, Jun Long, Yueyi Luo, and Chang Xu. Detecting any instruction-to-answer interaction relationship: Universal instruction-to-answer navigator for med-vqa. In *Forty-first International Conference on Machine Learning*, 2024.

[10] Ruixuan Xiao, Wentao Ma, Ke Wang, Yuchuan Wu, Junbo Zhao, Haobo Wang, Fei Huang, and Yongbin Li. Flowbench: Revisiting and benchmarking workflow-guided planning for llm-based agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10883–10900, 2024.

[11] Yiming Xiong, Jian Wang, Bing Li, Yuhan Zhu, and Yuqi Zhao. Self-organizing agent network for llm-based workflow automation. *arXiv preprint arXiv:2508.13732*, 2025.

[12] Mandar Kulkarni. Agent-s: Llm agentic workflow to automate standard operating procedures. *arXiv preprint arXiv:2503.15520*, 2025.

[13] Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. Odysseybench: Evaluating llm agents on long-horizon complex office application workflows. *arXiv preprint arXiv:2508.09124*, 2025.

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

[15] Junjie Chen, Haitao Li, Jingli Yang, Yiqun Liu, and Qingyao Ai. Enhancing llm-based agents via global planning and hierarchical execution. *arXiv preprint arXiv:2504.16563*, 2025.

[16] Nikolas Belle, Dakota Barnes, Alfonso Amayuelas, Ivan Bercovich, Xin Eric Wang, and William Wang. Agents of change: Self-evolving llm agents for strategic planning. *arXiv preprint arXiv:2506.04651*, 2025.

[17] Qiang Chen, Chun-Wun Cheng, Xiu Su, Hongyan Xu, Xi Lin, Shan You, Angelica I Aviles-Rivero, and Yi Chen. Legato: Good identity unlearning is continuous. *arXiv preprint arXiv:2601.04282*, 2026.

[18] Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. Agile: A novel reinforcement learning framework of llm agents. *Advances in Neural Information Processing Systems*, 37:5244–5284, 2024.

[19] Mingyue Cheng, Jie Ouyang, Shuo Yu, Ruiran Yan, Yucong Luo, Zirui Liu, Daoyu Wang, Qi Liu, and Enhong Chen. Agent-r1: Training powerful llm agents with end-to-end reinforcement learning. *arXiv preprint arXiv:2511.14460*, 2025.

[20] Yongcong Wang, Chengchao Shen, Guangwei Gao, Wei Wang, Pengwen Dai, Dianjie Lu, Guijuan Zhang, and Zhuoran Zheng. Libra-net: Lie-algebraic bilateral affine fields for real-time 4k video dehazing. *arXiv preprint arXiv:2605.11508*, 2026.

[21] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

[22] Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. Easytool: Enhancing llm-based agents with concise tool instruction. 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)*, pages 951–972, 2025.

[23] Yichao Cao, Qingfei Tang, Xiu Su, Song Chen, Shan You, Xiaobo Lu, and Chang Xu. Detecting any human-object interaction relationship: Universal hoi detector with spatial prompt learning on foundation models. *Advances in Neural Information Processing Systems*, 36:739–751, 2023.

[24] Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, et al. Skillcraft: Can llm agents learn to use tools skillfully? *arXiv preprint arXiv:2603.00718*, 2026.

[25] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6129–6139, 2025.

[26] Shalini Sudarsan, Advait Patel, Charit Upadhyay, Vaishnavi Gudur, and Prashanthi Matam. Secure prompt engineering patterns for cloud llm agents. In *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, pages 1–6. IEEE, 2026.

[27] Alexander D Kalian, Jaewook Lee, Stefan P Johannesson, Lennart Otte, Christer Hogstrand, and Miao Guo. Fine-tuning and prompt engineering of llms, for the creation of multi-agent ai for addressing sustainable protein production challenges. *arXiv preprint arXiv:2506.20598*, 2025.

[28] Yuxiang Zeng, Kun Xie, Yang Wang, Jigang Wen, Xiaocan Li, Guangxing Zhang, Gaogang Xie, and Jiannong Cao. Dta-gt: Direction-and topology-aware graph transformer for neural network representation learning. *arXiv preprint arXiv:2610.16822*, 2026.

[29] Nathaniel Mitrani Hadida, Sassan Bhanji, Cameron Tice, and Puria Radmard. Chain-of-thought obfuscation learned from output supervision can generalise to unseen tasks. *arXiv preprint arXiv:2601.23086*, 2026.

[30] Mingyu Chen, Jingkai Lin, Zhaojie Chu, Xiaofen Xing, Yirong Chen, and Xiangmin Xu. Catch: A novel data synthesis framework for high therapy fidelity and memory-driven planning chain of thought in ai counseling. *arXiv preprint arXiv:2509.25733*, 2025.

[31] Zhongze Wu, Xiu Su, Feng Yang, Shan You, Yueyi Luo, and Jun Long. Injection without distortion: Geometrically constrained knowledge enhancement for vision-language models. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

[32] Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*, 2025.

[33] Yifan Wei, Xiaoyan Yu, Yixuan Weng, Tengfei Pan, Angsheng Li, and Li Du. Autotir: Autonomous tools integrated reasoning via reinforcement learning. *arXiv preprint arXiv:2507.21836*, 2025.

[34] Xiaojie Guo, Yang Zhang, Bing Zhang, Ryo Kawahara, Mikio Takeuchi, and Yada Zhu. Unitoolbench: A benchmark for tool-augmented llms in cross-domain, universal task automation. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 4726–4736, 2026.

[35] Zhongze Wu, Xiu Su, Feng Yang, Dan Niu, Shan You, Yueyi Luo, and Jun Long. Unlearning without forgetting: Securely removing targeted concepts from large-scale vision-language open-vocabulary detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6271–6281, 2026.

[36] Qinzhuo Wu, Wei Liu, Jian Luan, and Bin Wang. Toolplanner: A tool augmented llm for multi granularity instructions with path planning and feedback. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18315–18339, 2024.

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

[38] Yuchen Shi, Siqi Cai, Zihan Xu, Yuei Qin, Gang Li, Hang Shao, Jiawei Chen, Deqing Yang, Ke Li, and Xing Sun. Flowagent: Achieving compliance and flexibility for workflow agents. *arXiv preprint arXiv:2502.14345*, 2025.

[39] Wei Yang, Jinwei Xiao, Hongming Zhang, Qingyang Zhang, Yanna Wang, and Bo Xu. Coarse-to-fine grounded memory for llm agent planning. *arXiv preprint arXiv:2508.15305*, 2025.

[40] Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, et al. Deepagent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*, pages 2219–2230, 2026.

[41] Chengying Huan, Ziheng Meng, Yongchao Liu, Zhengyi Yang, Yun Zhu, Yue Yun, Shipeng Li, Rong Gu, Xiabao Wu, Haitao Zhang, et al. Scaling graph chain-of-thought reasoning: A multi-agent framework with efficient llm serving. *arXiv preprint arXiv:2511.01633*, 2025.

[42] Xiang Li. Task planning and decision-making methods for intelligent agents based on large language models. In *Proceedings of the 4th International Conference on Artificial Intelligence and Intelligent Information Processing*, pages 817–822, 2025.

[43] Jiho Lee, Hayun Lee, Jonghyeon Kim, Kyungjae Lee, and Eunwoo Kim. Self-corrective task planning by inverse prompting with large language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11017–11023. IEEE, 2025.

[44] Yihong Guo, Dongqiangzi Ye, Sijia Chen, Anqi Liu, and Xianming Liu. Correctionplanner: Self-correction planner with reinforcement learning in autonomous driving. *arXiv preprint arXiv:2603.15771*, 2026.

[45] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. *arXiv preprint arXiv:2410.06153*, 2024.

[46] Caishen Zhou, Yihong Tang, Kehai Chen, Xuefeng Bai, Shuhan Qi, Li Shen, and Min Zhang. From fragmentation to systematic design: Architecting llm-based multi-agent systems. *Authorea Preprints*, 2026.

[47] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[48] Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jian-Guang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 496–507, 2025.

[49] 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.

[50] Yaswanth Chittepu, Raghavendra Addanki, Tung Mai, Anup Rao, and Branislav Kveton. Ml-tool-bench: Tool-augmented planning for ml tasks. *arXiv preprint arXiv:2512.00672*, 2025.

[51] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 3102–3116, 2023.

[52] Jize Wang, Zerun Ma, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. Gta: a benchmark for general tool agents. *Advances in Neural Information Processing Systems*, 37: 75749–75790, 2024.

Appendix Contents:

- A Proofs
- B Experimental Details
- C Additional Experiments
- D Broader Impact

A Proofs

This appendix provides the complete proofs for the theoretical results stated in Section 4. All notation follows the main text. At phase h, the context is $c_h = (x, \tau_{<h}, T_h)$, the generated plan is $P_h = (z_h^{(1)}, \ldots, z_h^{(L_h)})$, and the expert plan is $P_h^* = (y_h(s_1), \ldots, y_h(s_{L_h}))$. We reserve $U(P; c)$ for plan utility and $\mathcal{U}(c, t, o)$ for the deterministic context update.

Auxiliary notation. For two phase-h latent plans $P = (z^{(1)}, \ldots, z^{(L_h)})$ and $P' = (z'^{(1)}, \ldots, z'^{(L_h)})$, we use the plan metric:

$$\|P - P'\|_2 := \sum_{l=1}^{L_h} \|z^{(l)} - z'^{(l)}\|_2, \tag{22}$$

which is the plan metric used in Assumption 3. For a finite phase-h toolset, let $N_h := |T_h|$. All constants below are local to the finite horizon and to the compact semantic region visited by the train and test embeddings.

A.1 Proof of Proposition 1

Proposition 1 (Endpoint Error Bound). Let $z(1)$ be the flow endpoint and let $y(1)$ be the expert target anchor associated with the correct tool t^*. Let $z^*(s)$ denote the exact teacher trajectory induced by $u^*(\cdot, s \mid y(s))$ under the same initial prior sample as Eq. (6), so that $z^*(1) = y(1)$ by construction of the conditional flow target. Suppose that, on the compact region visited by the numerical solver, $u^*(\cdot, s \mid y(s))$ is K_u-Lipschitz in its first argument and

$$\|v_\phi(z(s), s \mid c_h) - u^*(z(s), s \mid y(s))\|_2 \le \delta \qquad \forall s \in [0, 1]. \tag{23}$$

Then the flow integration error

$$\eta_\phi := \mathbb{E}\|z(1) - y(1)\|_2 \tag{24}$$

satisfies $\eta_\phi \le C_{\text{flow}}\delta$, where $C_{\text{flow}} = (e^{K_u} - 1)/K_u$ if $K_u > 0$ and $C_{\text{flow}} = 1$ if $K_u = 0$. Consequently, for the MAP decoder induced by Eq. (2):

$$\mathbb{P}\{\hat{t}_{\text{MAP}} \neq t^*\} \le \min\left\{1, \frac{2\eta_\phi}{\Delta_{\text{dec}}}\right\} \le \min\left\{1, \frac{2C_{\text{flow}}\delta}{\Delta_{\text{dec}}}\right\}. \tag{25}$$

Thus the decoding error is controlled by the ratio between the flow integration error and the decoding margin.

Proof. Let $e(s) := \|z(s) - z^*(s)\|_2$. Since $z(0)$ and $z^*(0)$ use the same prior sample, $e(0) = 0$. The ODEs for $z(s)$ and $z^*(s)$ give

$$\frac{d}{ds}e(s) \le \|v_\phi(z(s), s \mid c_h) - u^*(z^*(s), s \mid y(s))\|_2 \tag{26}$$

$$\le \|v_\phi(z(s), s \mid c_h) - u^*(z(s), s \mid y(s))\|_2 + \|u^*(z(s), s \mid y(s)) - u^*(z^*(s), s \mid y(s))\|_2 \tag{27}$$

$$\le \delta + K_u e(s). \tag{28}$$

Gronwall's inequality yields:

$$\|z(1) - y(1)\|_2 = \|z(1) - z^*(1)\|_2 \le C_{\text{flow}}\delta. \tag{29}$$

Taking expectation gives $\eta_\phi \le C_{\text{flow}}\delta$.

It remains to connect endpoint error to discrete decoding. By Assumption 2, for every $t \in T_h \setminus \{t^*\}$

$$\|y(1) - e_{t^*}\|_2 \le \|y(1) - e_t\|_2 - \Delta_{\mathrm{dec}}. \tag{30}$$

Let $r := \|z(1) - y(1)\|_2$. The triangle inequality gives

$$\|z(1) - e_{t^*}\|_2 \le \|y(1) - e_{t^*}\|_2 + r \tag{31}$$

$$\le \|y(1) - e_t\|_2 - \Delta_{\mathrm{dec}} + r \tag{32}$$

$$\le \|z(1) - e_t\|_2 - \Delta_{\mathrm{dec}} + 2r. \tag{33}$$

Therefore, whenever $r < \Delta_{\mathrm{dec}}/2$, the correct tool t^* remains the unique nearest neighbor of $z(1)$ and the MAP decoder cannot make an error. Hence

$$\{\hat{t}_{\mathrm{MAP}} \neq t^*\} \subseteq \{\|z(1) - y(1)\|_2 \ge \Delta_{\mathrm{dec}}/2\}. \tag{34}$$

Applying Markov's inequality yields:

$$\mathbb{P}\{\hat{t}_{\mathrm{MAP}} \neq t^*\} \le \frac{2\mathbb{E}\|z(1) - y(1)\|_2}{\Delta_{\mathrm{dec}}} \le \frac{2C_{\mathrm{flow}}\delta}{\Delta_{\mathrm{dec}}}, \tag{35}$$

and clipping by 1 gives Eq. (25). □

When the pointwise velocity bound holds deterministically and $C_{\mathrm{flow}}\delta < \Delta_{\mathrm{dec}}/2$, Eq. (29) directly implies zero MAP decoding error; the probability bound covers random endpoint error when only its expectation is controlled.

Remark on stochastic decoding. Eq. (10) samples from the softmax decoder. Proposition 1 isolates the geometric error of the continuous endpoint and therefore uses the induced MAP decoder. The additional stochastic error introduced by finite temperature ϵ is bounded in Theorem 1 by the explicit $e^{-g_l/\epsilon}$ terms.

A.2 Proof of Theorem 1

Theorem 1 (Plan Utility Convergence). Let η_h bound the expected integrated velocity error and let each generated-anchor neighborhood retain a positive squared-distance decoder gap $g_l > 0$. If changing one decoded tool changes utility by at most B_U, then Eq. (19) holds.

Proof. We bound the three terms in Eq. (19) separately.

Step 1: continuous path approximation. Define

$$\eta_h := \mathbb{E}\left[\int_0^1 \|v_\phi(z_h(s), s \mid c_h) - u^*(z_h(s), s \mid y_h(s))\|_2 \, ds\right]. \tag{36}$$

Applying the same Gronwall argument as Proposition 1 up to each anchor time s_l gives

$$\mathbb{E}\|z_h^{(l)} - y_h(s_l)\|_2 \le C_{\mathrm{flow}}\eta_h, \qquad l = 1, \dots, L_h. \tag{37}$$

Assumption 3 therefore yields

$$\mathbb{E}|U(P_h^*; c_h) - U(P_h; c_h)| \le L_P L_h C_{\mathrm{flow}}\eta_h. \tag{38}$$

Moreover, Cauchy–Schwarz implies that the expected integrated error is at most the square root of the corresponding flow-matching mean-square error, so reducing $\mathcal{L}_{\mathrm{FM}}$ controls η_h.

Step 2: finite-temperature decoding. For anchor $z_h^{(l)}$ and its correct tool t_l^*, define

$$g_l := \inf_z \min_{t \in T_h \setminus \{t_l^*\}} \left(\|z - e_t\|_2^2 - \|z - e_{t_l^*}\|_2^2\right) > 0, \tag{39}$$

where the infimum is over the generated-anchor neighborhood. This is the explicit local-gap condition used by the theorem; Assumption 2 and sufficiently small anchor error are sufficient for such a neighborhood to exist. Equation (2) then gives

$$\Pr_\epsilon(\hat{t}_h^{(l)} \neq t_l^* \mid z_h^{(l)}) \le (N_h - 1)e^{-g_l/\epsilon}. \tag{40}$$

By bounded differences of utility and a union bound over the L_h positions,

$$|\mathbb{E}_\epsilon U(P_h; c_h) - U(P_h; c_h)| \le B_U(N_h - 1)\sum_{l=1}^{L_h} e^{-g_l/\epsilon}. \tag{41}$$

The stop decision is included in the same argument by treating STOP as an additional decoded label with its own positive gap.

Step 3: observation-induced context perturbation. For matched current context and tool, suppose the observation channel of the update satisfies

$$d_c\big(\mathcal{U}(c_h, \hat{t}_h, o), \mathcal{U}(c_h, \hat{t}_h, \hat{o})\big) \le \beta_o \|o - \hat{o}\|_2. \tag{42}$$

The context-Lipschitz part of Assumption 3 gives

$$\mathbb{E}|U(P_h; c_h) - U(P_h; \hat{c}_h)| \le L_c \beta_o \mathbb{E}\|o - \hat{o}\|_2. \tag{43}$$

Combining Eqs. (38), (41), and (43) proves Eq. (19). Each term vanishes under the limiting conditions stated in the theorem. □

A.3 Proof of Theorem 2

Theorem 2 (Unseen Tool Generalization). Let $\mathcal{T}_{\text{unseen}}$ be an unseen toolset whose embeddings lie on the semantic manifold $\mathcal{M}$. The expected generalization error satisfies

$$\mathbb{E}[\text{Err}_{\text{unseen}}] \le \mathbb{E}[\text{Err}_{\text{proxy}}] + C_{\mathcal{M}}(\delta_{\text{shift}} + \epsilon_{\text{cover}}), \tag{44}$$

where $\text{Err}_{\text{proxy}}$ is the error of the corresponding nearest training proxy, δ_{shift} is the semantic shift from the unseen region to the training region, and ϵ_{cover} is the covering radius of the training tool embeddings.

Proof. Let $\mathcal{M}_{\text{train}} \subset \mathcal{M}$ and $\mathcal{M}_{\text{unseen}} \subset \mathcal{M}$ denote the compact semantic regions occupied by training and unseen tools, respectively. Formalize the two quantities in Eq. (44) as

$$\delta_{\text{shift}} := \sup_{e \in \mathcal{M}_{\text{unseen}}} \inf_{\bar{e} \in \mathcal{M}_{\text{train}}} \|e - \bar{e}\|_2, \qquad \epsilon_{\text{cover}} := \sup_{\bar{e} \in \mathcal{M}_{\text{train}}} \min_{t \in \mathcal{T}_{\text{train}}} \|\bar{e} - e_t\|_2. \tag{45}$$

For any unseen tool $t_u \in \mathcal{T}_{\text{unseen}}$ with embedding e_{t_u}, choose $\bar{e} \in \mathcal{M}_{\text{train}}$ such that $\|e_{t_u} - \bar{e}\|_2 \le \delta_{\text{shift}}$ and then choose a training tool $\pi(t_u) \in \mathcal{T}_{\text{train}}$ such that $\|\bar{e} - e_{\pi(t_u)}\|_2 \le \epsilon_{\text{cover}}$. The triangle inequality yields:

$$\|e_{t_u} - e_{\pi(t_u)}\|_2 \le \delta_{\text{shift}} + \epsilon_{\text{cover}}. \tag{46}$$

Assumption 1 directly states that excess planning risk is $C_{\mathcal{M}}$-Lipschitz on the compact operational subset. Therefore replacing an unseen tool by its semantic proxy changes its error by at most

$$\text{Err}(t_u) \le \text{Err}(\pi(t_u)) + C_{\mathcal{M}} \|e_{t_u} - e_{\pi(t_u)}\|_2. \tag{47}$$

Combining Eqs. (46) and (47) gives

$$\text{Err}(t_u) \le \text{Err}(\pi(t_u)) + C_{\mathcal{M}}(\delta_{\text{shift}} + \epsilon_{\text{cover}}). \tag{48}$$

Taking expectation over $t_u \sim \mathcal{T}_{\text{unseen}}$ preserves the bound and proves Eq. (44). □

Interpretation. The proof separates two effects: the method must already predict the semantically nearest training proxy reliably, and transfer then pays an additional penalty determined by manifold shift and coverage. The result therefore does not imply zero unseen-tool error merely from a dense embedding space.

A.4 Proof of Proposition 2

Proposition 2 (Closed-loop Error Contraction). Under Assumption 4, the cumulative closed-loop context error satisfies Eq. (21), where E_{open} is the no-correction accumulation envelope formed from the same local errors $a_1, \ldots, a_{H-1}$.

Proof. Let c_h^* be the oracle context and set $e_h := d_c(c_h, c_h^*)$. By Assumption 4, the feedback-conditioned procedure satisfies

$$e_{h+1} \le \rho e_h + a_h, \qquad 0 \le \rho < 1, \tag{49}$$

with $e_1 = 0$ because the initial query is shared. Unrolling gives, for every $h \ge 2$,

$$e_h \le \sum_{i=1}^{h-1} \rho^{h-1-i} a_i. \tag{50}$$

Table 6: Structural distinction between the underlying τ^2-bench episode protocol and our plan-level supervision protocol.

Axis	**τ^2-bench**	**Our protocol**
Learning unit	Complete episode	Observed execution phase
Input	Task/environment state	Prefix, feedback, current tools
Target	Final task outcome/action	Complete remaining tool trajectory
One length-m trace	One episode	m prefix–suffix pairs
Execution	Closed-loop environment	Same environment; forward-only replan

Summing over phases and exchanging the finite sums yields

$$\begin{aligned} E_{\text{closed}} &:= \sum_{h=2}^{H} e_h \le \sum_{i=1}^{H-1} a_i \sum_{k=0}^{H-1-i} \rho^k \\ &= \sum_{i=1}^{H-1} a_i \frac{1-\rho^{H-i}}{1-\rho}. \end{aligned} \tag{51}$$

For $0 \le \rho < 1$, the geometric sum is at most its number of terms: $\sum_{k=0}^{H-1-i} \rho^k \le H - i$. Hence $E_{\text{closed}} \le \sum_{i=1}^{H-1}(H-i)a_i = E_{\text{open}}$. If $a_i > 0$ and $H - i \ge 2$, at least one coefficient in that geometric sum is strictly below one, so the inequality is strict. □

B Experimental Details

B.1 Benchmark Construction Details

Data Aggregation and Standardization. We build a unified, comprehensive tool-use benchmark by aggregating data from four prominent public sources: *τ^2-bench* (providing executable domains with real tools and databases), *ToolBench* (contributing large-scale API reasoning trajectories), *API-Bank* (ensuring broad schema coverage), and *GTA* (offering workflow-style task metadata). To ensure a rigorous evaluation protocol, we standardize the underlying tool schemas, task metadata, gold reasoning workflows, and explicitly define the dataset splits. In total, the unified benchmark features a dynamically evolving toolset of 3,930 unique tools and encompasses 6,865 complex multi-step tasks. To systematically evaluate semantic generalization, the toolset spans diverse functional phases (e.g., retrieval, verification, and database operations) and is rigorously partitioned, ensuring 599 tools are strictly unseen during testing.

State-Conditioned Closed-Loop Protocol. For executable evaluation, our primary experiments strictly adhere to a feedback-conditioned decoding formulation. At each planning step, given the user task, available toolsets, the executed action prefix, and our proposed compact state representation c_h, the model must either predict the subsequent discrete tool call or explicitly trigger the termination mechanism. Crucially, this setup operates under a strict forward-execution regime. The system has absolutely no access to oracle remaining plans, fundamentally preventing future trajectory leakage and ensuring a realistic assessment of the agent's dynamic replanning capabilities.

Functionally, τ^2-bench defines the interactive episode and judges task completion, whereas our transformation changes the supervised object at every phase: the observed prefix is paired with the complete unobserved suffix including STOP. After a real observation, the next phase produces a different pair and target; future actions never enter the conditioning prefix. Thus the contribution is precisely a *phase-conditioned prefix-to-remaining-trajectory supervision and evaluation protocol built on existing closed-loop environments*, not a claim to a new environment.

Executable Domains and Discriminative Signal. For physical execution, we strictly focus on the *telecom* and *retail* domains derived from *τ^2-bench*, curating a highly challenging test split of 140 closed-loop trajectories (63 *telecom* and 77 *retail*). The evaluation in these domains replays every prediction against real domain tools, initial states, and policies. While the telecom domain is largely saturated by current strong baselines, the retail split provides the most discriminative evaluation

signal. It explicitly challenges the model with complex, database-backed operational decisions—such as returning, exchanging, or modifying orders—that strictly depend on accurate intermediate state lookups.

B.2 Baseline Implementation Details

We compare methods under matched executor checkpoints, task splits, toolsets, horizons, termination rules, and forward-only execution. The configurations are:

- **Direct-LLM:** Serving as the direct-prompt baseline, this approach evaluates the inherent capabilities of instruction-tuned LLMs. The model directly predicts the next tool or the stop signal based purely on its prompt, without any benchmark-specific supervised fine-tuning (SFT) or reinforcement learning (RL) adaptation.
- **Standard SFT:** Serving as our clean supervised baseline, this model is fine-tuned to predict the next tool given the exact same context formulation as our method: the user task, available tools, the executed action prefix, and the compact state representation. Crucially, this baseline operates explicitly without the continuous Flow Matching (FM) prior, directly mapping the compact state to the next action.
- **RAG Planning:** Functioning as the retrieval-prior baseline, this method implements a retrieval-augmented LLM executor. For a given current state, it retrieves a semantically similar trajectory or state row from the training set. The retrieved next-tool decision or plan hint is then fed as an explicit prompt into the same LLM executor to guide the current prediction.
- **ToolRL:** Serving as the RL-route baseline, we directly implement the methodology proposed in ToolRL [14] and evaluate it on our newly constructed benchmark. To achieve this, our executable environment is specifically adapted to interface with the ToolRL and VERL training frameworks, and the policy is explicitly trained using Group Relative Policy Optimization (GRPO) to maximize trajectory-level rewards.
- **ToolPlanner, Agent-R1, and AutoTIR:** We additionally port the path-planning-with-feedback objective of ToolPlanner [36], the step-transition agentic-RL formulation of Agent-R1 [19], and the hybrid tool-selection reward of AutoTIR [33]. Only the action/output adapters are changed. All three retain their discrete interaction objectives and share FlowAgent's executor, split, context, toolset, horizon, termination rule, five seeds, and matched training-token/GPU-time budget.

B.3 Evaluation Metrics Formulation

To holistically assess the performance of the generated tool plans and their executable discrete translations, we deploy the following metric suite:

- **Overall Success & Retail Success:** The primary holistic metric evaluated over all trajectory rows. For next-action rows, success strictly requires both predicting the correct tool and generating an executable action payload. For stopping rows, it requires correctly triggering the termination mechanism. We separately report Retail Success to isolate the performance on the most challenging, dynamic database-backed domain.
- **Tool EM & Retail Tool EM:** This metric evaluates the pure planning policy. It measures the strict exact match of the predicted discrete tool identity $t \in T_h$ or the termination token. Similarly, Retail Tool EM isolates the model's ability to make the hardest DB-backed operation decisions without confounding execution errors.
- **Pred Exec:** A system-level execution metric evaluating whether the predicted discrete action payload can be successfully parsed and executed within the real tool environment (i.e., whether the state update operator $\mathcal{U}$ successfully returns a valid observation o_h).
- **Arg Value EM:** Evaluates the precision of the schema-driven structured grounding module. It is computed as the exact match of the instantiated argument key-value pairs against the gold execution trace. This metric is aggregated exclusively over action rows, as termination rows do not entail argument generation.

B.4 Configurations

Experiments on all datasets are conducted with the following hardware and software configuration:

- Operating System: Ubuntu 20.04.6 LTS
- CPU: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
- GPU: 3 $\times$ NVIDIA H800 GPUs, each with 80 GB of memory
- System RAM: 128 GB

Unless stated otherwise, the planner uses a 768-dimensional latent space, a fixed 16-step Euler ODE solver, decoder temperature $\epsilon = 0.1$, and five random seeds. The default frozen encoder is Qwen2.5-7B-Instruct: inputs are canonicalized text, truncated to 2,048 tokens, mean-pooled over final-layer hidden states using the attention mask, ℓ_2-normalized, and projected from 3,584 to 768 dimensions. Encoder-sensitivity experiments replace this component with the identically pooled Llama-3.2-3B-Instruct trunk or BERT-base-uncased; all encoders and tokenizers remain frozen.

C Additional Experiments

Diagnostic Baselines. We evaluate several heuristic diagnostic baselines in Table 7. Always Stop and Global Majority reach 0.2714 Overall Success, while Domain Majority identifies executable action types but rarely grounds valid arguments. State Key Majority reaches 0.7643 Tool EM yet only 0.6364 Pred Exec, showing that tool-name recovery alone does not imply executable task completion. Lexical Retrieval supplies a simple similarity-based reference for the learned retrieval baseline.

Table 7: Performance of diagnostic heuristic baselines. The results establish a basic performance floor and highlight the insufficiency of simple tool name matching.

Method	Overall Success ↑	Retail Success ↑	Tool EM ↑	Pred Exec ↑	Arg Value EM ↑
Always Stop	0.2714	0.0390	0.2714	-	-
Global Majority	0.2714	0.0390	0.2714	-	-
Domain Majority	0.4000	0.2727	0.4000	1.0000	0.1067
State Key Majority	0.5357	0.5195	0.7643	0.6364	0.2411
Lexical Retrieval	0.5286	0.5065	0.5643	0.7857	0.3083

Grounding Component Ablation. Our methodology regularizes semantic consistency between generated plans and the current observation context. The models frequently hallucinate invalid arguments even when successfully selecting the appropriate tool. To evaluate the engineering mechanisms enforcing this consistency, we conduct an ablation study in Table 8. We intentionally freeze the latent trajectory generation to maintain identical tool selection across all variants. Consequently, Tool EM remains constant across these variants. We evaluate the removal of specific grounding components including schema normalization, entity binding, field mapping, and state validation. The results reveal that state validation provides the primary defense against execution failures. Removing this validation step degrades the overall success from 0.7357 to 0.7071 and drastically reduces the retail success rate. Furthermore, the Pred Exec score drops to 0.8857. This degradation precisely mirrors the performance of the baseline lacking any validation. Other components maintain stable performance on the current evaluation subset. The result indicates that verifying generated parameters against environmental constraints is important for task completion.

Prior Injection Sensitivity. The methodology translates latent paths into discrete actions. Table 9 compares mechanisms for presenting latent priors to the decoder. Soft prefixes and structured text perform similarly in static pilots; we select structured text because it is auditable and uses the same interface during training and inference. We subsequently evaluate actual planning capabilities using dynamic loop experiments. The unconditioned flow prior yields a basic overall success rate of 0.6786. Integrating the prevailing context state creates the state conditioned prior and elevates overall success to 0.7357. Providing multiple candidates degrades performance to 0.7000. Injecting noise further

Table 8: Ablation study of individual grounding components. The tool selection process is fixed to isolate the contribution of specific execution validation mechanisms.

Method	Overall Success ↑	Retail Success ↑	Tool EM ↑	Retail Tool EM ↑	Pred Exec ↑	Arg Value EM ↑
No Grounding	0.7071	0.4675	0.7429	0.5325	0.8762	0.5455
Full Grounding	**0.7357**	**0.5195**	0.7429	0.5325	**0.9619**	**0.5613**
w/o Schema Nor.	0.7357	0.5195	0.7429	0.5325	0.9619	0.5613
w/o Entity Bind.	0.7357	0.5195	0.7429	0.5325	0.9619	0.5613
w/o Field Map.	0.7357	0.5195	0.7429	0.5325	0.9524	0.5613
w/o State Val.	0.7071	0.4675	0.7429	0.5325	0.8857	0.5455

reduces reliability. Ultimately, the state conditioned text prior provides the best balance of system transparency and reasoning accuracy.

Table 9: Sensitivity analysis of prior injection mechanisms. The table compares isolated pilot formats against dynamic loop performance to justify the selection of state conditioned text priors.

Method	Evaluation Mode	Overall Success ↑	Overall Tool EM ↑	Retail Success ↑	Retail Tool EM ↑	Arg Value EM ↑
Soft Prefix Pilot	Static Pilot	-	0.8916	-	0.5500	0.7698
Unstructured Text Pilot	Static Pilot	-	0.8916	-	0.5500	0.7599
Structured Text Pilot	Static Pilot	-	0.8916	-	0.5500	0.7867
Unconditioned Flow Prior	Dynamic Loop	0.6786	0.6857	0.4156	0.4286	0.5099
State Conditioned Prior	Dynamic Loop	**0.7357**	**0.7429**	**0.5195**	**0.5325**	**0.5613**
Multiple Candidate Prior	Dynamic Loop	0.7000	0.7000	0.4545	0.4545	0.4585
Noisy Prior Injection	Dynamic Loop	0.6714	0.6786	0.4026	0.4156	0.4901

Expanded Evaluation Protocol. The following experiments provide the complete numerical results corresponding to the new figures in the main paper. Unless noted otherwise, all methods within each comparison use identical executor checkpoints, splits, toolsets, interaction horizons, termination rules, forward-only execution, and five random seeds. Values with uncertainty are reported as mean $\pm$ sample standard deviation. We retain the same visual convention as the main paper: first-ranked results are green and underlined with a light-blue cell background, while second- and third-ranked results use blue and orange underlining.

Table 10: Matched comparison with recent RL tool-use planners on both submitted executors.

Executor	Method	Overall Succ.	Retail Succ.	Tool EM	Retail Tool EM	Pred Exec	Arg Value EM
Qwen2.5-7B	Standard SFT	0.6571	0.3766	0.6714	0.4026	0.9268	0.4387
	ToolRL	0.4786	0.5065	0.4929	0.5325	0.9643	0.5257
	ToolPlanner	0.6858	0.5352	0.6991	0.5583	0.9686	0.5512
	Agent-R1	0.7186	0.5528	0.7316	0.5750	0.9715	0.5684
	AutoTIR	0.6920	0.5262	0.7069	0.5504	**0.9764**	0.5562
	FlowAgent	**0.7643**	**0.5714**	**0.7745**	**0.5892**	0.9619	**0.5929**
Llama-3.2-3B	Standard SFT	0.6214	0.3117	0.6286	0.3247	0.9318	0.4345
	ToolRL	0.1357	0.1169	0.1500	0.1429	0.9362	0.1020
	ToolPlanner	0.6536	0.4545	0.6671	0.4805	0.9476	0.4708
	Agent-R1	0.6929	0.5192	0.7075	0.5452	0.9584	0.5352
	AutoTIR	0.6640	0.4808	0.6786	0.5061	**0.9657**	0.4979
	FlowAgent	**0.7571**	**0.5584**	**0.7714**	**0.5844**	0.9524	**0.5573**

RL Planner Analysis. Table 10 broadens the comparison from one RL baseline to three additional planner families. Agent-R1 is the strongest added baseline on task success and planning accuracy, whereas AutoTIR attains the highest Pred Exec. FlowAgent nevertheless ranks first on five of six metrics for both executors. Its advantage over Agent-R1 is larger on Llama-3.2-3B than on Qwen2.5-7B, indicating that the gain is not tied to a single backbone family; the exception on Pred Exec also shows that syntactic executability alone is insufficient for successful stateful reasoning.

Table 11: Larger-executor results on the original 140-trajectory executable split.

Executor	Method	Overall Succ.	Retail Succ.	Tool EM	Retail Tool EM	Pred Exec	Arg Value EM
Llama-3.1-8B	SFT	0.6764 ± 0.0138	0.4162 ± 0.0188	0.6893 ± 0.0129	0.4421 ± 0.0198	0.9447 ± 0.0078	0.4750 ± 0.0172
	ToolRL	0.4215 ± 0.0304	0.4032 ± 0.0292	0.4383 ± 0.0288	0.4295 ± 0.0310	0.9561 ± 0.0105	0.4306 ± 0.0279
	FlowAgent	**0.7890** ± 0.0101	**0.6106** ± 0.0145	**0.8000** ± 0.0094	**0.6235** ± 0.0154	**0.9685** ± 0.0068	**0.6264** ± 0.0128
Llama-3.1-70B	SFT	0.7571 ± 0.0096	0.5328 ± 0.0136	0.7690 ± 0.0091	0.5587 ± 0.0145	0.9665 ± 0.0057	0.5896 ± 0.0124
	ToolRL	0.6077 ± 0.0208	0.5712 ± 0.0221	0.6215 ± 0.0199	0.5970 ± 0.0234	0.9767 ± 0.0080	0.6008 ± 0.0206
	FlowAgent	**0.8508** ± 0.0074	**0.7013** ± 0.0108	**0.8618** ± 0.0069	**0.7142** ± 0.0116	**0.9833** ± 0.0050	**0.7081** ± 0.0097
Qwen2.5-14B	SFT	0.7002 ± 0.0115	0.4290 ± 0.0162	0.7142 ± 0.0108	0.4559 ± 0.0175	0.9446 ± 0.0068	0.4828 ± 0.0143
	ToolRL	0.5361 ± 0.0243	0.5584 ± 0.0267	0.5509 ± 0.0231	0.5842 ± 0.0280	**0.9720** ± 0.0089	0.5779 ± 0.0248
	FlowAgent	**0.8000** ± 0.0088	**0.6235** ± 0.0127	**0.8109** ± 0.0081	**0.6364** ± 0.0138	**0.9720** ± 0.0058	**0.6413** ± 0.0116
Qwen2.5-32B	SFT	0.7431 ± 0.0098	0.4939 ± 0.0141	0.7570 ± 0.0092	0.5198 ± 0.0150	0.9585 ± 0.0060	0.5487 ± 0.0127
	ToolRL	0.6145 ± 0.0195	0.6109 ± 0.0226	0.6294 ± 0.0187	0.6367 ± 0.0241	0.9795 ± 0.0076	0.6314 ± 0.0209
	FlowAgent	**0.8368** ± 0.0077	**0.6753** ± 0.0113	**0.8468** ± 0.0071	**0.6882** ± 0.0121	**0.9800** ± 0.0051	**0.6871** ± 0.0102

Larger-Executor Analysis. Table 11 evaluates four instruction-tuned executors on the same 140-trajectory split used by the submitted models. FlowAgent improves Overall Success over SFT by 0.1126, 0.0937, 0.0998, and 0.0937 on Llama-3.1-8B, Llama-3.1-70B, Qwen2.5-14B, and Qwen2.5-32B, respectively. Gains therefore persist within both model families as scale increases. ToolRL becomes stronger on larger backbones, especially on stateful retail metrics, but its higher standard deviations and lower task success indicate that backbone capacity alone does not replace complete-trajectory supervision.

Table 12: Tool EM regrouped by task source under unchanged checkpoints and splits.

Executor	Method	τ^2-bench	ToolBench	GTA
Qwen2.5-7B	Direct-LLM	0.1286	0.1742	0.1497
	Standard SFT	0.6714	0.6281	0.5962
	RAG Planning	0.5214	0.5796	0.5485
	ToolRL	0.4929	0.5480	0.5164
	FlowAgent	**0.7745**	**0.7284**	**0.7042**
Llama-3.2-3B	Direct-LLM	0.2075	0.2418	0.2195
	Standard SFT	0.6286	0.5847	0.5529
	RAG Planning	0.5500	0.5883	0.5612
	ToolRL	0.1524	0.2316	0.2074
	FlowAgent	**0.7714**	**0.7148**	**0.6895**

Cross-Source Analysis. Table 12 tests whether the planning gain is confined to the executable τ^2 domain. FlowAgent remains first on every source and backbone. Its smallest gain over the strongest source-specific baseline is 0.1003 Tool EM on Qwen2.5-7B/ToolBench, while its largest is 0.1428 on Llama-3.2-3B/τ^2-bench. RAG Planning is relatively competitive on ToolBench and GTA, where similar demonstrations aid retrieval, but FlowAgent's consistent margin shows that the result is not produced by one benchmark's schema distribution.

Table 13: Inference deployment cost on Qwen2.5-7B. NFE is the number of velocity-field function evaluations used by the ODE solver for one plan; autoregressive baselines do not invoke this solver.

Method	NFE	Planning median/p95 (ms)	End-to-end median/p95 (ms)	Peak memory (GiB)	Overall Succ.
Standard SFT	0	171/229	244/334	**15.4**	0.6571
ToolRL	0	176/237	250/343	15.6	0.4786
FlowAgent	16	194/259	270/369	16.1	**0.7643**

Inference-Cost Analysis. Table 13 separates training efficiency from inference cost. SFT and ToolRL require no ODE evaluations, whereas FlowAgent uses 16 NFE per generated plan. Relative to SFT, this adds 23 ms to median planning latency, 26 ms end to end, and 0.7 GiB of peak memory. The same comparison yields a 0.1072 increase in Overall Success, making the deployment trade-off explicit rather than treating the numerical solve as cost free.

Table 14: Prediction-target ablation on Qwen2.5-7B.

Planner	Prediction target per phase	Overall Succ.	Retail Succ.	Tool EM	Retail Tool EM
Standard SFT	One discrete next tool	0.6571 ± 0.0126	0.3766 ± 0.0174	0.6714 ± 0.0118	0.4026 ± 0.0192
One-step latent regression	One next-tool embedding	0.7149 ± 0.0121	0.4805 ± 0.0164	0.7286 ± 0.0113	0.5065 ± 0.0179
FlowAgent	Complete future latent trajectory	**0.7643** ± 0.0098	**0.5714** ± 0.0137	**0.7745** ± 0.0091	**0.5892** ± 0.0152

Prediction-Target Analysis. Table 14 answers whether CFM merely supplies a better embedding for the next action. One-step latent regression improves every metric over discrete SFT, confirming that semantic regression itself is useful. FlowAgent then improves Overall Success from 0.7149 to 0.7643 and Retail Success from 0.4805 to 0.5714. Because the encoder, executor, split, and decoding interface are matched, this remaining gain is attributable to supervising the complete future trajectory rather than only the next-tool embedding.

Table 15: Sensitivity to the number of predicted tools committed before replanning on Qwen2.5-7B.

Tools committed k	Overall Succ.	Retail Succ.	Tool EM	Arg Value EM
1 (FlowAgent)	**0.7643** ± 0.0098	**0.5714** ± 0.0137	**0.7745** ± 0.0091	**0.5929** ± 0.0124
2	0.7429 ± 0.0117	0.5455 ± 0.0156	0.7500 ± 0.0108	0.5718 ± 0.0142
3	0.7143 ± 0.0135	0.5065 ± 0.0178	0.7286 ± 0.0126	0.5484 ± 0.0163
Entire predicted plan	0.6714 ± 0.0159	0.4286 ± 0.0206	0.6857 ± 0.0148	0.5032 ± 0.0190

Replanning Analysis. Table 15 varies only how many decoded tools are committed before incorporating new feedback. All four metrics decline monotonically as k increases. Executing the entire provisional plan reduces Retail Success by 0.1428 and Arg Value EM by 0.0897 relative to $k = 1$, which is consistent with stale suffix decisions becoming invalid after database-changing observations. The result supports the combination of complete-trajectory generation for global structure and one-tool commitment for local correction.

Table 16: Llama-3.2-3B results on independently expanded executable test sets.

Test size	Method	Overall Succ.	Retail Succ.	Tool EM	Retail Tool EM	Pred Exec	Arg Value EM
300	SFT	0.6233 ± 0.0104	0.3156 ± 0.0137	0.6300 ± 0.0098	0.3299 ± 0.0145	0.9327 ± 0.0057	0.4382 ± 0.0128
	ToolRL	0.1433 ± 0.0247	0.1247 ± 0.0224	0.1567 ± 0.0238	0.1515 ± 0.0242	0.9371 ± 0.0085	0.1086 ± 0.0210
	FlowAgent	**0.7567** ± 0.0077	**0.5610** ± 0.0109	**0.7725** ± 0.0070	**0.5877** ± 0.0114	**0.9530** ± 0.0049	**0.5596** ± 0.0093
500	SFT	0.6198 ± 0.0080	0.3124 ± 0.0106	0.6272 ± 0.0076	0.3251 ± 0.0113	0.9315 ± 0.0044	0.4356 ± 0.0099
	ToolRL	0.1407 ± 0.0191	0.1204 ± 0.0174	0.1548 ± 0.0184	0.1466 ± 0.0188	0.9363 ± 0.0066	0.1057 ± 0.0162
	FlowAgent	**0.7582** ± 0.0059	**0.5591** ± 0.0084	**0.7711** ± 0.0054	**0.5850** ± 0.0088	**0.9525** ± 0.0038	**0.5579** ± 0.0072
800	SFT	0.6213 ± 0.0064	0.3131 ± 0.0084	0.6287 ± 0.0060	0.3268 ± 0.0089	0.9319 ± 0.0035	0.4364 ± 0.0078
	ToolRL	0.1395 ± 0.0151	0.1199 ± 0.0137	0.1536 ± 0.0145	0.1462 ± 0.0148	0.9366 ± 0.0052	0.1060 ± 0.0128
	FlowAgent	**0.7559** ± 0.0047	**0.5572** ± 0.0066	**0.7692** ± 0.0043	**0.5831** ± 0.0070	**0.9522** ± 0.0030	**0.5565** ± 0.0057

Test-Scale Analysis. Table 16 replaces the small 140-trajectory scope with three independently expanded sets. FlowAgent's Overall Success changes by at most 0.0023 across 300, 500, and 800 tests, and Tool EM changes by at most 0.0033. The ranking is unchanged on every metric. Standard deviations decrease with test size for all methods, while FlowAgent retains the smallest dispersion, showing that the original gain is not driven by a favorable 140-trajectory sample.

Table 17: Sensitivity to the frozen pretrained text encoder on Qwen2.5-7B execution.

Frozen encoder	Overall Succ.	Retail Succ.	Tool EM	Retail Tool EM	Pred Exec	Arg Value EM
Qwen2.5-7B-Instruct	**0.7643** ± 0.0098	**0.5714** ± 0.0137	**0.7745** ± 0.0091	**0.5892** ± 0.0152	**0.9619** ± 0.0061	**0.5929** ± 0.0124
Llama-3.2-3B-Instruct	0.7564 ± 0.0107	0.5584 ± 0.0154	0.7671 ± 0.0099	0.5766 ± 0.0165	0.9602 ± 0.0066	0.5830 ± 0.0135
BERT-base-uncased	0.7414 ± 0.0123	0.5325 ± 0.0178	0.7529 ± 0.0115	0.5506 ± 0.0191	0.9574 ± 0.0072	0.5652 ± 0.0156

Encoder Analysis. Table 17 measures how much performance is attributable to the frozen text encoder. Replacing Qwen with the Llama encoder changes Overall Success by only 0.0079 and Tool EM by 0.0074. The generic BERT-base control is lower than the instruction-tuned encoders but still exceeds Standard SFT by 0.0843 Overall-Success points and 0.0815 Tool-EM points. Thus pretrained semantic geometry affects the magnitude of the gain, but the method is not specific to the Qwen family or to a decoder-only encoder.

D Broader Impact

FlowAgent may improve the reliability and training efficiency of multi-tool agents by combining global trajectory supervision with feedback-conditioned execution. These capabilities can support reproducible research on planning under evolving toolsets. They also introduce familiar agentic risks: an incorrect but executable plan may modify external state, expose sensitive information, or amplify errors through high-impact APIs. Practical deployments should therefore retain access control, sandboxing, argument validation, audit logs, and human approval for consequential operations. Our current textual benchmarks do not establish safety in open-domain or safety-critical settings.

E Limitations and Future Work.

FlowAgent uses a fixed 16-step Euler solve, adding modest inference latency and memory; distillation or adaptive solvers may reduce this cost. The executable evidence remains textual and API-centered, so multimodal, open-domain, and safety-critical tools remain important tests. The bounds are explicitly conditional: out-of-manifold tools, collapsed decoder margins, or non-contractive/adversarial feedback can invalidate them. Expanded source, scale, encoder, and test-size studies probe robustness but do not remove these failure regimes.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[N/A]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: We have claimed the contributions and scope in lines 50-64.

   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: We have discussed the limitations in Section E.

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: [Yes]

   Justification: We provide the complete theoretical analyses in Section 4 and Appendix A.

   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: All the code for reproducing the experiments is made available in the anonymous github link.

   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: The data are publicly available. We properly introduce the data in Appendix B.

   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: We give experimental settings in Appendix B.

   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: [Yes]

   Justification: The experiments were conducted over 5 runs, and we present the averaged results.

   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: we provide the information on the computer resources in Appendix B.4: 3 $\times$ NVIDIA H800 GPUs each with 80 GB of memory, and Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz CPU.

   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: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   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: We have discussed social impacts in Appendix D.

    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: The paper poses no such 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: We have cited the original paper that produced the code package or dataset for the open-source frameworks.

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: [Yes]

Justification: We have introduced the proposed benchmark well.

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: The 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: Our 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: We have described the usage of LLMs.

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.