Agent-Dice: Disentangling Knowledge Updates via Geometric Consensus for Agent Continual Learning

Anonymous ACL submission

Abstract

Large Language Model (LLM)-based agents significantly extend the utility of LLMs by interacting with dynamic environments. However, enabling agents to continually learn new tasks without catastrophic forgetting remains a critical challenge, known as the stability–plasticity dilemma. We argue that this dilemma fundamentally arises from the failure to explicitly distinguish between common knowledge shared across tasks and conflicting knowledge introduced by task-specific interference. To address this, we propose Agent-Dice, a parameter fusion framework based on directional consensus evaluation. Concretely, Agent-Dice disentangles knowledge updates through a two-stage process: geometric consensus filtering to prune conflicting gradients, and curvature-based importance weighting to amplify shared semantics. We provide a rigorous theoretical analysis that establishes the validity of the proposed fusion scheme and offers insight into the origins of the stability–plasticity dilemma. Extensive experiments on GUI agents and tool-use agent domains demonstrate that Agent-Dice exhibits outstanding continual learning performance with minimal computational overhead and parameter updates.

1 Introduction

Recent advances in Large Language Models (LLMs) have spurred a paradigm shift in artificial intelligence, empowering agents with robust capabilities in reasoning (Plaat et al., 2024; Chen et al., 2025b), planning (Wei et al., 2025; Huang et al., 2024), and decision-making (Sun et al., 2025; Huang et al., 2025). These agents expand the boundaries of the capabilities of LLMs' by deploying them in dynamic real-world environments, specifically by operating graphical user interfaces (GUIs) (Zhang et al., 2024a) or utilizing tools (Li, 2025).

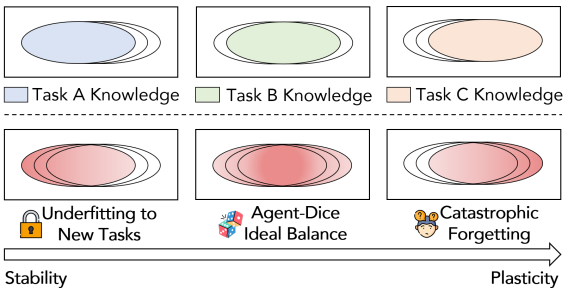

Figure 1: The stability-plasticity dilemma in agent continual learning (from Task A to Task C). Agent-Dice finds a balance between stability and plasticity by learning common knowledge.

To further enhance their capabilities, it is critical for agents to possess continual learning (CL) (Zheng et al., 2025; Gao et al., 2025) skills, enabling continuous self-iteration and adaptation to novel tasks without retraining from scratch. However, agent continual learning is fundamentally hindered by the stability-plasticity dilemma (Robins, 1995). As shown in Figure 1, during the adaptation process for new tasks, an agent faces a critical trade-off: overemphasizing stability hampers new learning, whereas excessive plasticity induces catastrophic forgetting.

Achieving the stability-plasticity dilemma essentially requires a precise disentanglement of knowledge updates. Ideally, agents are expected to minimize the interference from conflicting knowledge, while efficiently identifying and reinforcing the learning of common knowledge shared between new and old tasks to maintain plasticity. Existing approaches primarily tackle this via two paradigms: incorporating external memory modules (Ouyang et al., 2025; Zhang et al., 2025a) or conducting continuous iterative training (Zhang et al., 2025c; Muppidi et al., 2024). However, these methods often fail to effectively distinguish between common and conflicting

knowledge, inevitably leading to information loss or interference during the learning process.

To overcome these limitations, we propose Agent-Dice, a principled parameter fusion framework based on **Di**rectional **C**onsensus **E**valuation. Agent-Dice integrates task vectors from diverse tasks onto the original agent via a two-stage process: (1) Geometric Consensus Filtering, which prunes conflicting updates to preserve stability; and (2) Curvature-based Importance Weighting, which amplifies shared consensus directions to enhance plasticity.

Extensive experiments in the domains of GUI agent and tool-use agent demonstrate that Agent-Dice outperforms traditional continual learning paradigms with extremely low time overhead and minimal parameter update costs. And we further validate the rationality and effectiveness of Agent-Dice through ablation studies, model similarity analysis, and overhead evaluation.

To summarize, our contributions are four-fold:

(i) We identify that the stability–plasticity dilemma in continual learning for LLM-based agents largely arises from the failure to explicitly distinguish between common and conflicting knowledge during the learning process.

(ii) We propose Agent-Dice, a novel parameter fusion framework that integrates geometric consensus filtering with curvature-based importance weighting, enabling effective multi-task continual learning for LLM agents.

(iii) We present a theoretical analysis that proves the validity of the Agent-Dice parameter fusion scheme, while also providing new insights into the root causes of the stability–plasticity dilemma in agent continual learning.

(iv) Extensive experiments across both GUI agent and tool-use domains demonstrate that Agent-Dice exhibits outstanding continual learning performance with minimal computational overhead and parameter updates.

2 Related Work

In this section, we first review recent advances in LLM-based agents, focusing on two representative agent paradigms studied in this work: GUI agents and tool-use agents. We then summarize prior efforts on continual learning for LLMs.

2.1 LLM Agent

Recent advances in LLMs have empowered LLM-based agents to interact with complex environments by leveraging their capabilities reasoning (Plaat et al., 2024), planning (Wei et al., 2025), and decision-making (Sun et al., 2025). One representative line of research focuses on GUI agents (Tang et al., 2025b; Zhang et al., 2024a), which operate smart devices through human-like interactions and adapt to new tasks via large-scale pre-training (Wang et al., 2025a; Ye et al., 2025), supervised fine-tuning (Ma et al., 2024; Zhang and Zhang, 2024), and reinforcement learning (Tang et al., 2025a; Lu et al., 2025b; Luo et al., 2025; Liu et al., 2025b; Xu et al., 2025; Bai et al., 2024; Wang et al., 2025b). Another important direction is tool-use agents, which extend LLM capabilities by integrating external tools and APIs to perform complex reasoning and execution (Schick et al., 2023; Qin et al., 2023; Liu et al., 2024, 2025a; Zhang et al., 2025b; Patil et al., 2025; Barres et al., 2025; Chen et al., 2025a). Despite their strong performance, most existing agents are adapted to new domains through sequential fine-tuning or continual updates, which often leads to interference between previously acquired and newly learned skills. This limitation highlights the need for more principled continual learning mechanisms tailored to LLM agents.

2.2 LLM Continual Learning

To enable LLMs to better adapt to new tasks, existing studies on continual learning for LLMs have explored several main directions. These include regularization-based methods that constrain parameter updates or feature representations (Kirkpatrick et al., 2017; Zenke et al., 2017), approaches that store and replay a subset of previous data (Rebuffi et al., 2017; Hou et al., 2019), and architecture-based strategies that introduce task-specific modules or models (Schwarz et al., 2018; Yan et al., 2021). More recently, rehearsal-free methods have gained attention by leveraging parameter-efficient strategies for continual fine-tuning of pre-trained models (Wang et al., 2022; Tang et al., 2023; Wang et al., 2023). However, most existing approaches are developed in the context of traditional LLM tasks, while continual learning for LLM agents in complex and dynamic environments presents additional challenges that have not yet been fully addressed.

3 Agent-Dice

In this section, we present **Agent-Dice**, a theoretically grounded parameter fusion framework. We first provide a theoretical support (complete proof provided in Appendix A) of our method using a optimization perspective. We then introduce the detailed implementation pipeline of Agent-Dice.

3.1 Theoretical Support

Let $\boldsymbol{\theta}^* \in \mathbb{R}^d$ be the optimal parameters on the pre-trained manifold. We consider a multi-task setting with K tasks, where each task k is associated with a loss function $\mathcal{L}_k : \mathbb{R}^d \to \mathbb{R}$. The fine-tuned parameter vector for task k is denoted by $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{\text{pre}} + \boldsymbol{\tau}_k$, where $\boldsymbol{\tau}_k$ represents the task-specific displacement vector.

Our goal is to find a fusion policy Φ such that the fused parameter $\boldsymbol{\theta}_{\text{fused}} = \Phi(\{\boldsymbol{\theta}_k\}_{k=1}^K)$ minimizes the worst-case approximation error relative to the Pareto-optimal solution of the joint loss $\mathcal{L}_{\text{total}}(\boldsymbol{\theta}) = \sum_{k=1}^K \mathcal{L}_k(\boldsymbol{\theta})$. We analyze the fusion process through three theoretical lenses: linear approximation, variance reduction via consensus, and maximum entropy weight assignment.

Parameter Space Linearization. First, we establish the validity of the linear combination form applied in an element-wise manner (meaning each parameter is combined individually). We rely on the assumption that the pre-trained model lies in a *linear mode connectivity* basin (Mirzadeh et al., 2021), a phenomenon widely observed in large-scale deep learning models.

Theorem 1 (First-Order Manifold Aggregation)**.** *Assume that for a local neighborhood around $\boldsymbol{\theta}_{pre}$, the loss function $\mathcal{L}_k$ is approximately linear with respect to $\boldsymbol{\tau}_k$. Let $\mathbf{W} \in \mathbb{R}^{d\times K}$ be a weighting matrix where $\sum_{k=1}^K w_{k,i} = 1$. The update rule $\boldsymbol{\theta}_{new} = \boldsymbol{\theta}_{pre} + \sum_{k=1}^K \mathbf{w}_k \odot \boldsymbol{\tau}_k$ approximates a single gradient descent step on a surrogate multi-task objective $\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \sum_{k=1}^K \mathbf{w}_k^\top \mathcal{L}_k(\boldsymbol{\theta})$.*

Proof. Using a first-order Taylor expansion, $\mathcal{L}_k(\boldsymbol{\theta}_{\text{pre}} + \boldsymbol{\tau}) \approx \mathcal{L}_k(\boldsymbol{\theta}_{\text{pre}}) + \nabla\mathcal{L}_k(\boldsymbol{\theta}_{\text{pre}})^\top\boldsymbol{\tau}$. Since $\boldsymbol{\tau}_k$ is obtained via SGD, $\boldsymbol{\tau}_k \propto -\nabla\mathcal{L}_k(\boldsymbol{\theta}_{\text{pre}})$. The fused update becomes: $\Delta\boldsymbol{\theta} \propto -\nabla\left(\sum_{k=1}^K \mathbf{w}_k^\top \mathcal{L}_k(\boldsymbol{\theta}_{\text{pre}})\right)$. This confirms that the fusion rule minimizes the joint loss. □

Noise Suppression via Geometric Consistency. Multi-task fusion often suffers from gradient interference. We model the task vectors as noisy estimators of a shared latent descent direction to justify the necessity of consensus-based filtering.

Definition (Interference Model). For a parameter j, let the true descent sign be $s_j^* \in \{-1, +1\}$. We assume that the sign of the k-th task update, $s_{k,j} = \text{sgn}(\tau_{k,j})$, follows a Bernoulli distribution with success probability $p > 0.5$, i.e., $P(s_{k,j} = s_j^*) = p$.

Theorem 2 (Consensus-Induced Variance Reduction)**.** *Let $\mathcal{S}_j$ be the set of tasks with consistent signs for parameter j. If outlier tasks (where $s_{k,j} \neq s_j^*$) are excluded from aggregation, the probability of update error decays exponentially with the size of the consensus set $|\mathcal{S}_j|$, strictly outperforming standard averaging.*

Proof. Let X be the number of consistent tasks. By Hoeffding's inequality, the probability that the majority vote is incorrect is bounded by:

$$P(\text{error}) \leq \exp(-2|\mathcal{S}_j|(p-0.5)^2). \quad (1)$$

Standard averaging includes the minority set, which effectively reduces the margin $p - 0.5$ or introduces destructive interference, thus increasing the error bound. Filtering ensures that the update remains within the cone of the true gradient. □

Saliency Maximization via Boltzmann Distribution. Finally, we formalize the assignment of scalar weights. We posit that the magnitude $|\tau_{k,j}|$ serves as a proxy for the local sensitivity (curvature) of the loss landscape, and thus cast the weight selection as a Maximum Entropy problem.

Theorem 3 (Optimal Weighting under Saliency Constraints)**.** *Let $u_{k,j} = |\tau_{k,j}|$ denote the saliency of parameter j for task k. The probability distribution $\mathbf{w}_j$ that maximizes the entropy $H(\mathbf{w}_j)$ subject to the constraint of matching the expected saliency is the Boltzmann distribution:*

$$w_{k,j} = \frac{\exp(\beta u_{k,j})}{\sum_{m\in\mathcal{S}_j}\exp(\beta u_{m,j})}, \quad (2)$$

where β is the inverse temperature parameter.

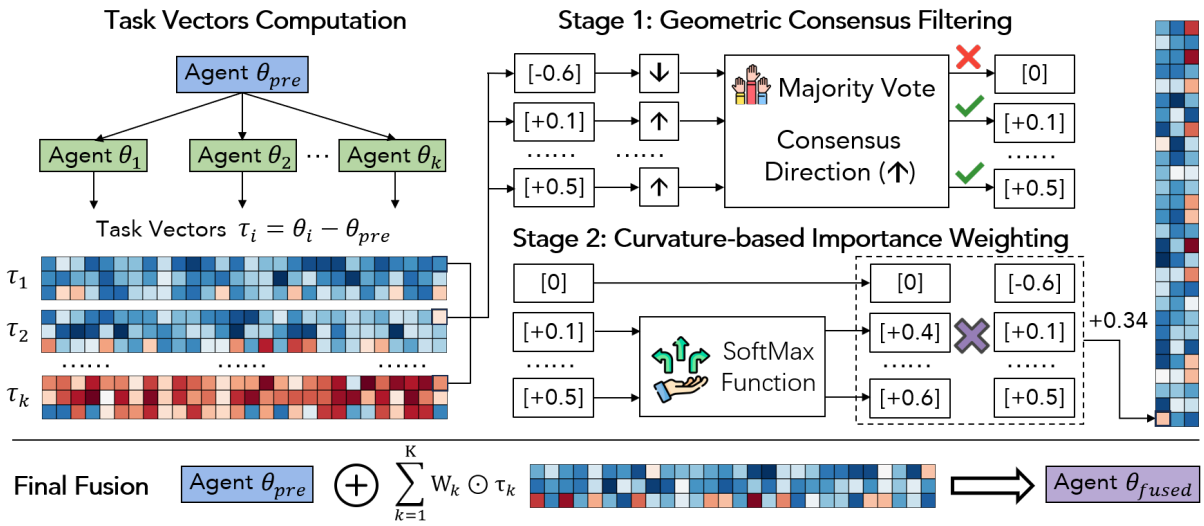

Figure 2: **The Agent-Dice Parameter Fusion Pipeline.** Task vectors $\boldsymbol{\tau}_k$ undergo a two-stage aggregation policy: *Geometric Consensus Filtering* for variance reduction via outlier pruning, and *Curvature-based Importance Weighting* for entropy maximization based on parameter saliency. The final refined update is added to $\boldsymbol{\theta}_{\text{pre}}$.

Proof. We formulate the Lagrangian $\mathcal{L} = -\sum_k w_{k,j} \log w_{k,j} + \lambda(\sum_k w_{k,j} u_{k,j} - C) + \gamma(\sum_k w_{k,j} - 1)$. Setting the partial derivative $\frac{\partial \mathcal{L}}{\partial w_{k,j}} = 0$ yields $\log w_{k,j} = \lambda u_{k,j} + \gamma - 1$, which implies $w_{k,j} \propto \exp(\lambda u_{k,j})$. This derivation confirms that Softmax is the least biased distribution given the saliency magnitudes. □

3.2 Method: Directional Consensus Evaluation for Parameter Fusion

Guided by the theoretical support, we formalize the Agent-Dice algorithm. The method operates element-wise on the agent parameters to construct a fused update. To provide a clear overview, we first present the *general formulation* of the fusion process. Supported by Theorem 1, the final fused parameter vector $\boldsymbol{\theta}_{\text{fused}}$ is obtained by aggregating the candidate task vectors $\{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_K\}$ weighted by a dynamic consensus matrix $\mathbf{W}_k$:

$$\boldsymbol{\theta}_{\text{fused}} = \boldsymbol{\theta}_{\text{pre}} + \sum_{k=1}^{K} \mathbf{W}_k \odot \boldsymbol{\tau}_k, \tag{3}$$

where $\odot$ denotes the Hadamard product (element-wise multiplication), and $\mathbf{W}_k \in \mathbb{R}^d$ represents the element-wise importance weight vector associated with the agent parameters $\boldsymbol{\theta}_k$ learned from the k-th task. The core innovation of Agent-Dice lies in the specific construction of $\mathbf{W}_k$, which is determined through a two-stage process: geometric consensus filtering and curvature-based importance weighting for continual learning.

Stage 1: Geometric Consensus Filtering. Supported by Theorem 2, we first identify the dominant optimization direction to construct the active set for each parameter. This step acts as a binary mask, filtering out outlier updates that contradict the manifold consensus.

Let $\tau_{k,i}$ denote the update value of the k-th agent for the i-th parameter, where $i \in \{1, \ldots, d\}$. We define the sign indicator $s_{k,i} \in \{0, 1\}$ as $s_{k,i} = 1(\tau_{k,i} \geq 0)$. The consensus score for the i-th parameter is given by the positive vote count $V_i = \sum_{k=1}^{K} s_{k,i}$. The active set $\mathcal{S}_i$ for this specific parameter is determined by a majority threshold δ (in this paper, δ is set to $K/2$):

$$\mathcal{S}_i = \begin{cases} \{k \mid s_{k,i} = 1\}, & \text{if } V_i > \delta, \\ \{k \mid s_{k,i} = 0\}, & \text{if } V_i < K - \delta, \\ \{1, \ldots, K\}, & \text{otherwise.} \end{cases} \tag{4}$$

Only agents belonging to $\mathcal{S}_i$ are considered eligible to contribute to the fusion of the i-th parameter; effectively, weights for agents $k \notin \mathcal{S}_i$ will be forced to zero.

Stage 2: Curvature-based Importance Weighting. After filtering, supported by Theorem 3, we determine the specific values of the weights $w_{k,i} \in \mathbf{W}_k$ for the eligible agents. Large update magnitudes $|\tau_{k,i}|$ typically indicate high confidence or traversal through steep gradients in the loss landscape (high curvature) for that specific parameter. To prioritize these informative features, we employ a masked Softmax function to

normalize these magnitudes within the active set for each parameter i:

$$w_{k,i} = \begin{cases} \frac{\exp(|\tau_{k,i}|)}{\sum_{j \in \mathcal{S}_i} \exp(|\tau_{j,i}|)}, & \text{if } k \in \mathcal{S}_i, \\ 0, & \text{if } k \notin \mathcal{S}_i. \end{cases} \quad (5)$$

This weighting scheme ensures that the final model trajectory follows the consensus of the most confident agents locally for each parameter, effectively neutralizing catastrophic forgetting caused by conflicting tasks.

Final Fusion. By substituting the computed element-wise weights $w_{k,i}$ back into the general formulation in Equation 3, we obtain the final updated parameter vector $\boldsymbol{\theta}_{\text{fused}}$. This aggregation effectively integrates the directional consensus with magnitude-based confidence. Specifically, for every parameter i, the update becomes a weighted sum $\sum_{k \in \mathcal{S}_i} w_{k,i} \tau_{k,i}$, where the contribution of conflicting agents is nullified. Consequently, the fused model updates strictly along the manifold direction determined by the majority, while the step size is adaptively governed by the agents exhibiting the strongest local feature response. This ensures the global optimization trajectory balances stability and plasticity via consensus filtering and curvature weighting, respectively.

4 Experiments

In this section, we validate the effectiveness of Agent-Dice in two domains: GUI agent and tool-use agent. First, we will briefly introduce the implementation, and then present our and analyse our main results.

4.1 Implementation

Dataset. For the GUI agent domain, we choose three popular benchmarks: AITZ (Zhang et al., 2024b), AndroidControl (Li et al., 2024), and GUI-Odyssey (Lu et al., 2025a). For the tool-use agent domain, we chose ToolACE (Liu et al., 2025a) as the dataset and partitioned it into four subsets based on the greedy algorithm (Appendix C) according to the minimum tool overlap.

Evaluation Protocol. We simulate the learning paradigm of continual learning agents by incrementally adding new knowledge to supervised fine-tuning the agent. For the GUI agent domain, we add new benchmarks to train the agent. For the tool-use agent domain, we gradually add new subsets to train the agent. We report the results in the zero-shot setting, the setting where each task is trained individually, and the setting where tasks are trained sequentially and continuously. More detailed evaluation protocol can be found in Appendix B.

Metrics. Our core reported metric is the average Z-score (AvgZ) from multi-task learning evaluation. Let $\{M_i\}_{i=1}^N$ be the metric scores across N tasks. We compute: Z-score$(M_i) = \frac{M_i - \mu_i}{\sigma_i}$, AvgZ $= \frac{1}{N} \sum_{i=1}^N$ Z-score(M_i). where μ_i and σ_i are the mean and standard deviation from baseline models on task i.

- For the GUI agent domain, we also report the action type accuracy (Type), step-wise success rate (SR), and trajectory success rate (TSR), where TSR equals 1 only if the SR for every step in the trajectory is 1.

- For the tool-use agent domain, we report the rate of predicting the correct tool function name (Func) and the rate of correctly predicting both the tool function name and its parameters (Full).

Models. For the GUI agent domain, we select OS-Atlas-Pro-7B (Wu et al., 2025) and Qwen3-VL-8B (Bai et al., 2025) for experimentation. This is because OS-Atlas-Pro-7B is a model specifically designed for the GUI agent domain, while Qwen3-VL-8B is a general-purpose model that has undergone training in the GUI agent domain, making both representative choices. For the tool-use agent domain, we select Qwen3-8B (Yang et al., 2025) and Llama-3.1-8B (Dubey et al., 2024) for experimentation. Qwen3-8B inherently possesses tool-use capabilities, allowing it to achieve good tool-use performance in a zero-shot setting, whereas Llama-3.1-8B lacks tool-use capabilities. Thus, these two models represent distinct scenarios.

4.2 Main Results

The experimental results, as shown in Tables 1-4, lead to the following key findings:

(i) In both the GUI agent domain and the agent tool-use domain, Agent-Dice achieves the highest AvgZ. This indicates that Agent-Dice outperforms traditional lifelong-learning agents in incremental learning, demonstrating its effectiveness.

Method	AITZ			AndroidControl			GUI-Odyssey			AvgZ
	Type	SR	TSR	Type	SR	TSR	Type	SR	TSR	
Zero-Shot	63.41	45.08	0.20	73.12	47.14	13.63	74.22	54.71	0.60	-0.38
Learn from AITZ	**75.63**	**59.92**	**6.13**	61.82	36.70	7.54	83.33	60.19	0.54	0.09
Learn from AndroidControl	61.34	30.85	0.00	**85.25**	**57.95**	**20.31**	74.68	40.80	0.24	-0.26
Learn from GUI-Odyssey	63.16	37.67	0.20	69.03	36.79	6.29	**90.74**	**76.06**	**4.62**	-0.17
CL from AITZ and AndroidControl	65.81	37.41	0.00	84.49	57.47	19.40	73.85	39.95	0.30	-0.14
CL from all three	65.78	42.05	0.04	73.56	43.48	9.57	90.69	75.79	4.44	0.14
Agent-Dice (Ours)	74.72	57.10	2.37	80.03	51.42	14.42	89.27	72.28	2.10	**0.73**

Table 1: Experiment results of Agent-Dice in the GUI agent domain, with OS-Atlas-Pro-7B as the base model. The best results are highlighted in **bold**, while the second-best are underlined and the third-best are underwaved.

Method	AITZ			AndroidControl			GUI-Odyssey			AvgZ
	Type	SR	TSR	Type	SR	TSR	Type	SR	TSR	
Zero-Shot	56.81	41.14	0.99	73.90	52.18	13.76	67.68	44.08	0.42	-0.02
Learn from AITZ	**74.72**	**58.23**	**5.34**	58.00	30.23	3.80	66.56	37.17	0.00	-0.03
Learn from AndroidControl	49.12	20.56	0.00	**83.21**	**61.92**	**20.12**	65.22	28.04	0.00	-0.30
Learn from GUI-Odyssey	58.06	32.01	0.20	66.78	31.07	4.72	**88.54**	**67.42**	**1.86**	0.10
CL from AITZ and AndroidControl	59.45	31.44	0.00	82.90	61.91	18.94	64.99	28.66	0.00	-0.06
CL from all three	60.03	35.97	0.59	72.47	34.23	5.05	88.09	63.83	0.48	0.01
Agent-Dice (Ours)	68.28	47.49	0.40	79.39	41.50	8.45	81.40	54.27	0.42	**0.29**

Table 2: Experiment results of Agent-Dice in the GUI agent domain, with Qwen3-VL-8B as the base model. The best results are highlighted in **bold**, while the second-best are underlined and the third-best are underwaved.

(ii) Overall, the AvgZ of the zero-shot setting in each experimental group is negative, indicating that training can indeed enhance the overall capability of the agent. Moreover, the learn from all setting is often not the second-best, suggesting that when continuously learning new knowledge, the agent's old knowledge may be affected, leading to a decline in overall capability.

(iii) For the GUI agent domain, due to the significant differences in APP data across different datasets, a clear catastrophic forgetting problem is observed. When a agent encounters knowledge from GUI-Odyssey, it exhibits noticeable forgetting of the knowledge related to AITZ and Androidcontrol APPs. With Agent-Dice, compared to learning from all three datasets, the performance on AITZ and Androidcontrol is greatly improved while only a slight decline in metrics is observed on GUI-Odyssey.

(iv) For the agent tool-use domain, Agent-DiCE yields more pronounced gains via common knowledge reinforcement and noise filtering, given minor cross-tool learning mechanism disparities in those tasks. It achieves top metrics across nearly all subsets and works effectively for both tool-use-capable (Qwen3-8B) and tool-use-less-capable (Llama-3.1-8B) models.

5 Further Analysis

In this section, we first validate the rationale of Agent-Dice's design through an ablation study. Then, we differentiate Agent-Dice from direct sequential learning of task knowledge via a model similarity analysis. Finally, we demonstrate the lightweight nature of Agent-Dice through an overhead evaluation.

5.1 Ablation Study

We conduct an ablation study on the two stages of Agent-Dice: geometric consensus filtering and curvature-based importance weighting. For stage 1 ablation, we remove the voting-based weighting mechanism and instead assign uniform weights to all new task vectors during training. For stage 2 ablation, we discard the final importance weighting step, i.e., no curvature-based reweighting is applied to the updates. We report the Full metric on the tool-use agent domain and the SR metric on the

Method	Subset 0		Subset 1		Subset 2		Subset 3		AvgZ
	Func	Full	Func	Full	Func	Full	Func	Full	
Zero-Shot	**99.64**	81.85	99.26	85.66	98.52	83.85	99.28	86.36	-1.42
Learn from Subset 0	98.93	85.96	**99.63**	88.83	99.26	88.81	**100.0**	91.35	0.27
Learn from Subset 1	99.29	86.82	**99.63**	87.52	98.89	89.36	99.64	90.52	0.13
Learn from Subset 2	98.57	84.59	**99.63**	87.71	97.79	87.52	98.57	90.35	-1.02
Learn from Subset 3	**99.64**	85.79	**99.63**	88.83	99.26	88.62	99.64	89.85	0.28
CL from Subset 0 & 1	99.29	86.30	**99.63**	89.57	99.26	**89.54**	99.64	91.18	0.44
CL from Subset 0, 1 & 2	99.29	87.16	**99.63**	89.01	99.26	88.07	**100.0**	91.68	0.48
CL from all Subsets	**99.64**	86.13	98.88	88.64	**99.63**	88.44	99.64	90.68	0.06
Agent-Dice (Ours)	99.29	**87.33**	**99.63**	**90.69**	99.26	89.36	**100.0**	**92.18**	**0.79**

Table 3: Experiment results of Agent-Dice in the tool-use domain, with Qwen3-8B as the base model. The best results are highlighted in **bold**, while the second-best are underlined and the third-best are underwaved.

Method	Subset 0		Subset 1		Subset 2		Subset 3		AvgZ
	Func	Full	Func	Full	Func	Full	Func	Full	
Zero-Shot	13.93	5.82	14.87	6.89	13.28	5.14	10.49	10.32	-2.81
Learn from Subset 0	**98.57**	79.45	97.77	82.31	**98.52**	83.67	**98.57**	85.19	0.45
Learn from Subset 1	88.57	72.95	91.45	73.37	92.62	75.05	89.96	75.87	0.13
Learn from Subset 2	96.79	74.14	96.65	76.54	97.05	74.50	96.77	80.37	0.29
Learn from Subset 3	95.36	78.25	**98.88**	83.05	97.05	80.55	95.34	83.86	0.39
CL from Subset 0 & 1	95.71	79.62	97.40	83.24	96.31	81.10	96.42	83.19	0.40
CL from Subset 0, 1 & 2	93.57	74.14	93.31	76.72	93.73	78.53	93.91	82.20	0.26
CL from all Subsets	95.00	78.94	94.42	82.50	95.94	81.10	97.49	83.19	0.38
Agent-Dice (Ours)	97.50	**84.25**	98.14	**85.47**	**98.52**	**85.69**	98.21	**87.02**	**0.51**

Table 4: Experiment results of Agent-Dice in the tool-use domain, with Llama-3.1-8B as the base model. The best results are highlighted in **bold**, while the second-best are underlined and the third-best are underwaved.

GUI agent domain.

Results are illustrated in Figures 3-4. Under the stage 1 ablation setting, the agent fails to learn precise common knowledge across tasks, highlighting the critical role of geometric consensus filtering. Under the stage 2 ablation setting, the agent's performance drops sharply in both the GUI agent and tool-use agent domains. This degradation is caused by the absence of curvature-based importance weighting, which otherwise constrains the update magnitude; without this constraint, the knowledge updates become excessively large and unstable. Overall, these results demonstrate that both stages are indispensable for stable and effective knowledge integration in Agent-Dice.

5.2 Model Similarity Analysis

As shown in Figures 5-6, we evaluate the model similarity between the Agent-Dice fused model, the base model, and models continuously trained on individual datasets. We utilize KL divergence as the metric, where a lower value indicates greater similarity. The experimental results of the model similarity analysis are presented as heatmaps.

Overall, in the GUI agent domain and the tool-use agent domain, when a base model is trained on more tasks, the KL divergence with the base model increases. This is because the agent continuously learns new knowledge to adapt to new tasks.

In the GUI agent domain, catastrophic forgetting is more pronounced. So Agent-Dice strikes a balance in parameter shifts across the three datasets. It attains the best performance with only a marginal increase in parameter modification compared to single-dataset training.

In the tool-use agent domain, Agent-Dice exhibits the highest similarity to Qwen3-8B. This suggests that Agent-Dice effectively captures the common knowledge intrinsic to the tool-use domain, achieving superior performance with minimal parameter deviation.

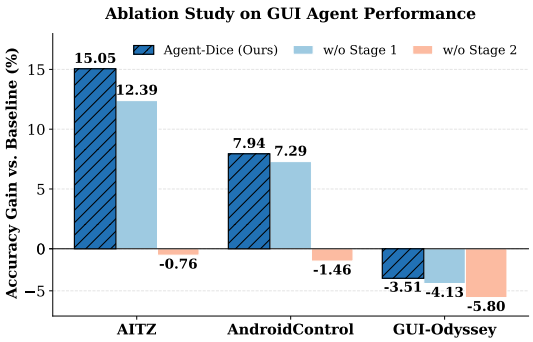

Figure 3: Ablation study on GUI Agent tasks.

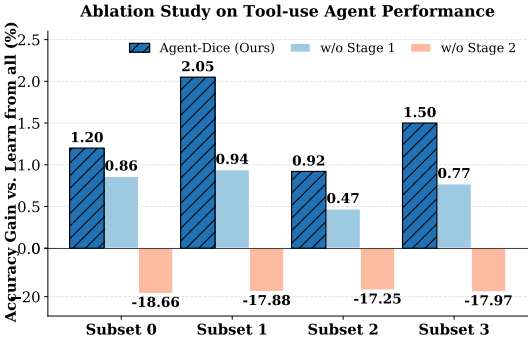

Figure 4: Ablation study on Tool-use Agent tasks.

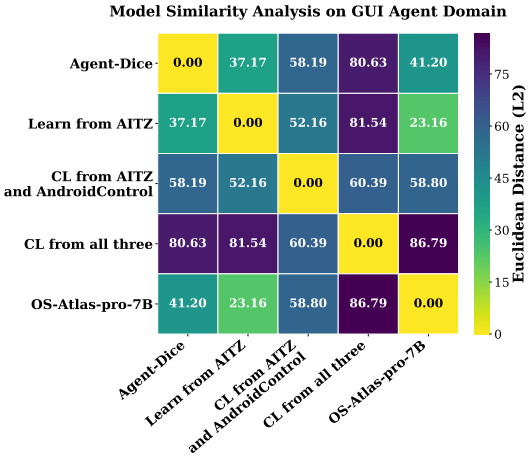

Figure 5: Model similarity analysis in the GUI agent domain. The similarity between Agent-Dice and OS-Atlas-Pro-7B is only marginally lower than that of models trained on a single dataset.

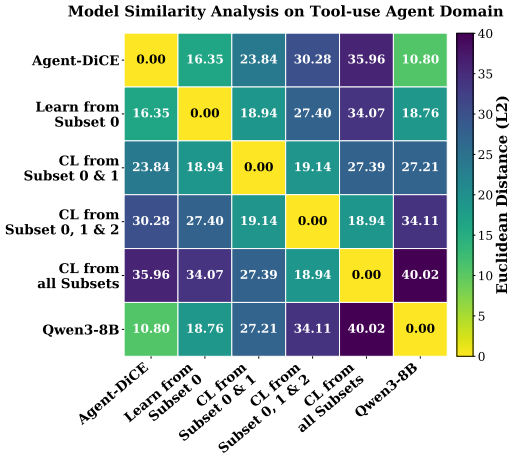

Figure 6: Model similarity analysis in the tool-use agent domain. Agent-Dice exhibits the highest similarity to Qwen3-8B.

5.3 Overhead Evaluation

To analyze the additional time overhead introduced by Agent-Dice, we conduct an overhead evaluation experiment. We test different base models in the tool-use agent domain and the GUI agent domain, and report results for scenarios using only the GPU and only the CPU, respectively. On average, Agent-Dice requires only about one minute of GPU usage or about ten minutes of CPU usage to complete the task.

The statistics are reported in Table 5. When using a GPU, the overhead of Agent-Dice ranges only from 61.84 to 88.52 seconds. Even without a GPU, when using only the CPU, the overhead of Agent-Dice does not exceed one minute. Compared to the training process that takes several hours or dozens of hours, the time overhead introduced by Agent-Dice is negligible. This fully demonstrates that Agent-Dice is a lightweight and efficient agent continual learning solution.

Model	Domain	GPU Time (s)	CPU Time (s)
OS-Atlas-Pro-7B	GUI	73.93	559.47
Qwen3-VL-8B	GUI	83.88	463.97
Qwen3-8B	Tool-use	61.84	461.03
LLaMA-3.1-8B	Tool-use	88.52	1049.82
Average	-	77.04	633.57

Table 5: Computation time comparison across different domains using Agent-Dice with GPU or CPU.

6 Conclusion

In this work, we identify that the stability–plasticity dilemma in continual learning for LLM-based agents largely arises from the failure to explicitly distinguish between common and conflicting knowledge during the learning process. To address the challenge, we present Agent-Dice, a novel parameter fusion framework designed to resolve the stability-plasticity dilemma in agent continual learning with minimal computational overhead and parameter updates. Further, we demonstrate the rationality and effectiveness of Agent-Dice through ablation study, model similarity analysis, and overhead evaluation.

Limitations

While Agent-Dice has been validated through extensive experiments across different backbone models in the GUI agent domain and the tool-use agent domain, the evaluation centers on a limited set of representative agent scenarios. However, this limitation lies in the scope of empirical evaluation rather than in the design of the proposed method itself. Future work may explore additional agent domains and task settings to further examine the generality and applicability of Agent-Dice under more diverse and realistic conditions.

References

Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

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

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, Wulong Liu, Xinzhi Wang, Defu Lian, Baoqun Yin, Yasheng Wang, and Wu Liu. 2025a. Acebench: Who wins the match point in tool usage? *Preprint*, arXiv:2501.12851.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025b. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839.

Jincai Huang, Yongjun Xu, Qi Wang, Qi Cheems Wang, Xingxing Liang, Fei Wang, Zhao Zhang, Wei Wei, Boxuan Zhang, Libo Huang, and 1 others. 2025. Foundation models and intelligent decision-making: Progress, challenges, and perspectives. *The Innovation*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154.

Xinzhe Li. 2025. A review of prominent paradigms for llm-based agents: Tool use, planning (including rag), and feedback learning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9760–9779.

Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, and 1 others. 2025a. Toolace: Winning the points of llm function calling. In *The Thirteenth International Conference on Learning Representations*.

Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. 2025b. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh Murthy, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. 2024. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Preprint*, arXiv:2406.18518.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. 2025a. Guiodyssey: A comprehensive dataset for cross-app

gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22404–22414.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025b. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*.

Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. 2025. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9097–9110.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. 2021. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*.

Aneesh Muppidi, Zhiyu Zhang, and Heng Yang. 2024. Fast trac: A parameter-free optimizer for lifelong reinforcement learning. *Advances in Neural Information Processing Systems*, 37:51169–51195.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, and 1 others. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*.

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

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. 2024. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Anthony Robins. 1995. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR.

Chuanneng Sun, Songjun Huang, and Dario Pompili. 2025. Llm-based multi-agent decision-making: Challenges and future directions. *IEEE Robotics and Automation Letters*.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, and 1 others. 2025a. Gui-g2: Gaussian reward modeling for gui grounding. *arXiv preprint arXiv:2507.15846*.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, and 1 others. 2025b. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*.

Yu-Ming Tang, Yi-Xing Peng, and Wei-Shi Zheng. 2023. When prompt-based incremental learning does not meet strong pretraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1706–1716.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, and 1 others. 2025a. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*.

Liyuan Wang, Jingyi Xie, Xingxing Zhang, Mingyi Huang, Hang Su, and Jun Zhu. 2023. Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. *Advances in Neural Information Processing Systems*, 36:69054–69076.

Taiyi Wang, Zhihao Wu, Jianheng Liu, Jianye HAO, Jun Wang, and Kun Shao. 2025b. Distrl: An asynchronous distributed reinforcement learning framework for on-device control agent. In *The Thirteenth International Conference on Learning Representations*.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *Proceedings of*

the IEEE/CVF conference on computer vision and pattern recognition, pages 139–149.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025. Plangenllms: A modern survey of llm planning capabilities. *arXiv preprint arXiv:2502.11221*.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and 1 others. 2025. Os-atlas: Foundation action model for generalist gui agents. In *The Thirteenth International Conference on Learning Representations*.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. 2025. Mobilerl: Online agentic reinforcement learning for mobile gui agents. *arXiv preprint arXiv:2509.18119*.

Shipeng Yan, Jiangwei Xie, and Xuming He. 2021. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3014–3023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, and 1 others. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, and 1 others. 2024a. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024b. Android in the zoo: Chain-of-action-thought for gui agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025a. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Kangning Zhang, Wenxiang Jiao, Kounianhua Du, Yuan Lu, Weiwen Liu, Weinan Zhang, Lei Zhang, and Yong Yu. 2025b. Looptool: Closing the data-training loop for robust llm tool calls. *arXiv preprint arXiv:2511.09148*.

Zhi Zhang, Chris Chow, Yasi Zhang, Yanchao Sun, Haochen Zhang, Eric Hanchen Jiang, Han Liu, Furong Huang, Yuchen Cui, and OSCAR HERNAN MADRID PADILLA. 2025c. Statistical guarantees for lifelong reinforcement learning using pac-bayes theory. In *International Conference on Artificial Intelligence and Statistics*, pages 5050–5058. PMLR.

Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3132–3149.

Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. 2025. Lifelong learning of large language model based agents: A roadmap. *arXiv preprint arXiv:2501.07278*.

A Proof Datails

In this section, we provide more detailed proofs for Theorem 1, Theorem 2, and Theorem 3.

A.1 Detailed Proof of Theorem 1

Our goal is to prove the update rule $\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{pre}} + \sum_{k=1}^{K} \mathbf{w}_k \odot \boldsymbol{\tau}_k$ approximates a single gradient descent step on a surrogate multi-task objective $\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \mathbf{w}_k^\top \mathcal{L}_k(\boldsymbol{\theta})$.

By the linear mode connectivity assumption:

$$\mathcal{L}_k(\boldsymbol{\theta}+\boldsymbol{\delta}) = \mathcal{L}_k(\boldsymbol{\theta})+\nabla\mathcal{L}_k(\boldsymbol{\theta})^\top\boldsymbol{\delta}+\mathcal{O}(\|\boldsymbol{\delta}\|^2). \quad (6)$$

For small $\|\boldsymbol{\delta}\|$:

$$\mathcal{L}_k(\boldsymbol{\theta}+\boldsymbol{\delta}) \approx \mathcal{L}_k(\boldsymbol{\theta}) + \nabla\mathcal{L}_k(\boldsymbol{\theta})^\top\boldsymbol{\delta}. \quad (7)$$

Each fine-tuning update $\boldsymbol{\tau}_k$ is obtained via gradient descent with learning rate η:

$$\boldsymbol{\tau}_k = -\eta\nabla\mathcal{L}_k(\boldsymbol{\theta}). \quad (8)$$

Define scalar weights $w_k \in \mathbb{R}$ with $\sum_{k=1}^{K} w_k = 1$. The fused update is:

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{pre}} + \sum_{k=1}^{K} w_k\boldsymbol{\tau}_k. \quad (9)$$

Substituting (8) into (9):

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{pre}} - \eta\sum_{k=1}^{K} w_k\nabla\mathcal{L}_k(\boldsymbol{\theta}_{\text{pre}}). \quad (10)$$

Define the surrogate loss:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \sum_{k=1}^{K} w_k \mathcal{L}_k(\boldsymbol{\theta}). \tag{11}$$

From (10) and (11):

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta} - \eta \nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}). \tag{12}$$

Thus, the fused update equals one gradient descent step on $\tilde{\mathcal{L}}$ with step size η.

And we define:

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta} + \sum_{k=1}^{K} \mathbf{w}_k \odot \boldsymbol{\tau}_k. \tag{13}$$

From (8) and (13):

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta} - \eta \sum_{k=1}^{K} \mathbf{w}_k \odot \nabla \mathcal{L}_k(\boldsymbol{\theta}). \tag{14}$$

From (12) and (14):

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \mathbf{w}_k^\top \mathcal{L}_k(\boldsymbol{\theta}). \tag{15}$$

A.2 Detailed Proof of Theorem 2

Our goal is to show that consensus-based filtering yields an update direction whose error probability decays exponentially with the number of consistent tasks, and improves over standard averaging.

Consider a fixed parameter dimension j. Let $s_j^* \in \{-1, +1\}$ denote the true descent direction. For each task k, define the signed update

$$s_{k,j} = \operatorname{sgn}(\tau_{k,j}). \tag{16}$$

Assume

$$P(s_{k,j} = s_j^*) = p, \quad p > \frac{1}{2}, \tag{17}$$

and $\{s_{k,j}\}_{k=1}^{K}$ are independent.

Let $\mathcal{S}_j \subseteq \{1, \ldots, K\}$ denote the consensus set after filtering, and let $m = |\mathcal{S}_j|$. Define the random variable

$$X = \sum_{k \in \mathcal{S}_j} \mathbb{I}[s_{k,j} = s_j^*], \tag{18}$$

with expectation

$$\mathbb{E}[X] = mp. \tag{19}$$

An update error occurs if the aggregated direction disagrees with s_j^*, i.e.,

$$X \leq \frac{m}{2}. \tag{20}$$

Applying Hoeffding's inequality yields

$$P\Big(X \leq \frac{m}{2}\Big) \leq \exp\big(-2m(p - 0.5)^2\big). \tag{21}$$

Thus, the error probability decays exponentially with the consensus size m.

For standard averaging, all K tasks are aggregated, including those with $s_{k,j} \neq s_j^*$. This corresponds to an effective success probability $\tilde{p} \leq p$, yielding

$$P_{\text{avg}}(\text{error}) \geq \exp\big(-2K(\tilde{p} - 0.5)^2\big), \quad \tilde{p} < p. \tag{22}$$

Comparing with (21), consensus-based filtering achieves a strictly tighter error bound.

A.3 Detailed Proof of Theorem 3

Our goal is to derive the optimal scalar weighting $\mathbf{w}_j$ that maximizes entropy under expected saliency and normalization constraints.

For a fixed parameter dimension j, let

$$u_{k,j} = |\tau_{k,j}|$$

denote the saliency associated with task k. We seek a distribution

$$\mathbf{w}_j = \{w_{k,j}\}_{k \in \mathcal{S}_j} \tag{23}$$

that maximizes the entropy

$$H(\mathbf{w}_j) = -\sum_{k \in \mathcal{S}_j} w_{k,j} \log w_{k,j}, \tag{24}$$

subject to the constraints

$$\sum_{k \in \mathcal{S}_j} w_{k,j} u_{k,j} = C, \tag{25}$$

$$\sum_{k \in \mathcal{S}_j} w_{k,j} = 1. \tag{26}$$

We form the Lagrangian

$$\begin{aligned} \mathcal{L} = & -\sum_k w_{k,j} \log w_{k,j} \\ & + \lambda \left(\sum_k w_{k,j} u_{k,j} - C \right) \\ & + \gamma \left(\sum_k w_{k,j} - 1 \right). \end{aligned} \tag{27}$$

Taking the partial derivative with respect to $w_{k,j}$ and setting it to zero:

$$\frac{\partial \mathcal{L}}{\partial w_{k,j}} = -\log w_{k,j} - 1 + \lambda u_{k,j} + \gamma = 0. \tag{28}$$

Solving for $w_{k,j}$ yields

$$\log w_{k,j} = \lambda u_{k,j} + \gamma - 1, \tag{29}$$

or equivalently,

$$w_{k,j} \propto \exp(\lambda u_{k,j}). \tag{30}$$

Enforcing the normalization constraint (26), we obtain

$$w_{k,j} = \frac{\exp(\lambda u_{k,j})}{\sum_{m \in \mathcal{S}_j} \exp(\lambda u_{m,j})}. \tag{31}$$

By defining $\beta = \lambda$, the optimal solution corresponds to the Boltzmann (Softmax) distribution, completing the proof.

B Experimental Details

In this section, we provide a comprehensive description of the experimental setup used to evaluate Agent-Dice. We detail the implementation configurations, action and output formats, and model selections across two representative domains: the GUI agent domain and the tool-use agent domain. All experiments are conducted under consistent training and evaluation protocols to ensure fair and reproducible comparisons.

B.1 Implementation Details

All experiments were performed using 1200 hours of 80GB GPU computing resources. We conducted training with llama-factory, setting a learning rate of 1e-5 for 3 epochs when using OS-Atlas-pro-7B as the base model in the GUI agent domain, and a learning rate of 1e-5 for 2 epochs when using Qwen3-VL-8B as the base. For the tool-use agent domain, a learning rate of 1.0e-5 was applied for 3 epochs when using both Qwen3-8B and Llama-3.1-8B as base models.

B.2 Output Format

For the GUI agent domain, we follow the common action space used by existing GUI agents, as shown in Table 6. During evaluation, we adhere to the assessment methods of existing works: for actions with coordinates such as CLICK and LONG_PRESS, a relative error of less than 14% is considered correct. For TYPE actions, an F1 score greater than 0.5 is required to be counted as correct. In all other cases, exact matching is necessary for correctness. And TSR for a task will be 1 only if SR for every single frame within that task is 1.

For the tool-use agent domain, we specify the selectable tools and parameter descriptions in the input prompt. The agent directly outputs a list of tool calls, where each element includes the function name along with the corresponding parameter names and values.

B.3 Model details

Our experiments are conducted in two domains: the GUI agent domain and the tool-use agent domain. In the GUI agent domain, we consider both models specialized for GUI manipulation and general-purpose models equipped with GUI interaction capabilities. To ensure broad and representative evaluation, we select one example from each category, namely OS-Atlas-Pro-7B and Qwen3-VL-8B, for our experiments.

In the tool-use agent domain, not all models are able to follow prompts to invoke tools under a zero-shot setting. Therefore, we choose Qwen3-8B and Llama-3.1-8B, which demonstrate zero-shot tool-use capability, for evaluation.

C How to partition the ToolACE dataset

To ensure a balanced distribution of tool capabilities across different partitions, we employ the splitting strategy outlined in Algorithm 1.

In the assignment phase, the ToolACE dataset is first shuffled to eliminate distributional bias. We then adopt a greedy allocation approach where each sample is assigned to the subset that minimizes a joint objective: the increment of unseen tools (to promote tool concentration) and the current subset size (to ensure load balancing). This tool-aware mechanism effectively prevents the fragmentation of tool occurrences, allowing each subset to specialize in specific functional domains while maintaining uniform data volume. Subsequently, we execute a density-based intra-subset split to construct robust training and evaluation sets. Within each assigned subset, samples are sorted by tool usage density in descending order; the top portion (determined by ratio r) is selected for training to maximize the model's exposure to complex tool-use scenarios. For the test set, we identify the tool coverage of the training partition and organize the remaining samples based on their tool novelty relative to the training data.

As evidenced in Table 7, the diagonal entries exhibit significantly higher tool overlap ($25.8\% \sim$

Action Type	Action Description	Action Format
CLICK	Click at the specified position.	CLICK <point>[[x-axis, y-axis]]</point>
TYPE	Enter specified text at the designated location.	TYPE [input text]
SCROLL	Scroll in the specified direction.	SCROLL [UP/DOWN/LEFT/RIGHT]
PRESS_BACK	Press a back button to navigate to the previous screen.	PRESS_BACK
PRESS_HOME	Press a home button to navigate to the home page.	PRESS_HOME
ENTER	Press the enter button.	ENTER
OPEN_APP	Open the specified application.	OPEN_APP [app_name]
WAIT	Wait for the screen to load.	WAIT
LONG_PRESS	Long press at the specified position.	LONG_PRESS <point>[[x-axis, y-axis]]</point>
COMPELTE	Indicate the task is finished.	COMPELTE
IMPOSSIBLE	Indicate the task is impossible.	IMPOSSIBLE

Table 6: Action space in our GUI agent domain experiment.

Algorithm 1: ToolACE Split Strategy

INPUT : Dataset $\mathcal{D}$, number of subsets M, training ratio r
OUTPUT : Subsets $\{\mathcal{D}_m^{\text{train}}, \mathcal{D}_m^{\text{test}}\}_{m=1}^{M}$

Phase 1: Tool-Aware Assignment
Initialize $\mathcal{D}_m \leftarrow \emptyset, \mathcal{T}_m \leftarrow \emptyset$ for $m \in \{1, \ldots, M\}$
foreach $x \in \texttt{Shuffle}(\mathcal{D})$ **do**
  $\mathcal{T}(x) \leftarrow$ extract tools from x
  $m^* \leftarrow \arg\min_m (|\mathcal{T}(x) \setminus \mathcal{T}_m|,\ |\mathcal{D}_m|)$
  $\mathcal{D}_{m^*} \leftarrow \mathcal{D}_{m^*} \cup \{x\}; \quad \mathcal{T}_{m^*} \leftarrow \mathcal{T}_{m^*} \cup \mathcal{T}(x)$
end
Phase 2: Intra-Subset Splitting
for $m = 1$ **to** M **do**
  $\mathcal{D}_m \leftarrow \texttt{SortDesc}(\mathcal{D}_m, \text{key} = |\mathcal{T}(x)|)$
  $N_{\text{train}} \leftarrow \lfloor r \cdot |\mathcal{D}_m| \rfloor$
  $\mathcal{D}_m^{\text{train}} \leftarrow \mathcal{D}_m[1 : N_{\text{train}}]$
  $\mathcal{T}_m^{\text{train}} \leftarrow \bigcup_{x \in \mathcal{D}_m^{\text{train}}} \mathcal{T}(x)$
  $\mathcal{D}_{\text{rem}} \leftarrow \mathcal{D}_m \setminus \mathcal{D}_m^{\text{train}}$
  $\mathcal{D}_m^{\text{test}} \leftarrow \texttt{SortAsc}(\mathcal{D}_{\text{rem}}, \text{key} = |\mathcal{T}(x) \setminus \mathcal{T}_m^{\text{train}}|)$
end
return $\{\mathcal{D}_m^{\text{train}}, \mathcal{D}_m^{\text{test}}\}_{m=1}^{M}$

	$\mathcal{D}_0^{\text{test}}$	$\mathcal{D}_1^{\text{test}}$	$\mathcal{D}_2^{\text{test}}$	$\mathcal{D}_3^{\text{test}}$
$\mathcal{D}_0^{\text{train}}$	**103 (29.6 %)**	4 (1.2 %)	7 (2.1 %)	6 (1.8 %)
$\mathcal{D}_1^{\text{train}}$	8 (2.3 %)	**94 (29.1 %)**	5 (1.5 %)	6 (1.8 %)
$\mathcal{D}_2^{\text{train}}$	11 (3.2 %)	2 (0.6 %)	**106 (31.8 %)**	5 (1.5 %)
$\mathcal{D}_3^{\text{train}}$	6 (1.7 %)	4 (1.2 %)	4 (1.2 %)	**84 (25.8 %)**

Table 7: Tool overlap statistics between training and testing subsets. Diagonal entries (in bold) indicate intra-subset overlap, while off-diagonal entries represent cross-subset overlap.

31.8%) compared to the minimal cross-subset leakage ($< 3.2\%$) in off-diagonal entries. This distinct boundary confirms that our splitting strategy effectively localizes tool usage within each partition, ensuring that each subset specializes in a distinct functional domain while maintaining strong consistency between its training and testing distributions.

D Case Study

In this section, we provide examples to show how GUI agents and the tool-use agent work.

D.1 GUI Agent Case Study

The goal of a GUI agent is to automatically execute instructions on smart terminals by simulating human operations, following user-given commands. As shown in Figure 7, a user needs guidance for a trip to Bangkok, Thailand, and then requires a flight ticket. The GUI agent first searches for guidance. It clicks the Threads APP icon on the main interface, enters the app, clicks the search button, and inputs text to search for relevant content. After finding relevant guidance, it selects the option to return to the main interface using the home button and proceeds to search for flight tickets. Finally, the task is terminated because no matching flight tickets are found.

D.2 Tool-Use Agent Case Study

The goal of tool-use agent is to enable the agent to automatically determine whether it needs to call a tool based on user-given instructions, decide which tool to call, and output the correct function name, parameter names, and parameter values. As shown in Figure 8, in this prompt, the user asks the agent to inform them which songs were at the top of the Billboard Holiday 100 chart in 2025. This falls outside the agent's intrinsic knowledge, so it needs to call a tool. Among the tools provided by the user, the Holiday 100 Songs API is the most suitable for the user's instruction. Therefore, this function should be called. The Holiday 100 Songs API has two parameters: `year` and `artist`. The `artist` parameter is irrelevant to the user's query, so the agent's final response is `[Holiday 100 Songs API(year=2025)]`.

Task: I'm looking for guidance on how to Use Threads and Agoda to explore the optimal time for visiting Bangkok, Thailand, and then proceed to book a flight.

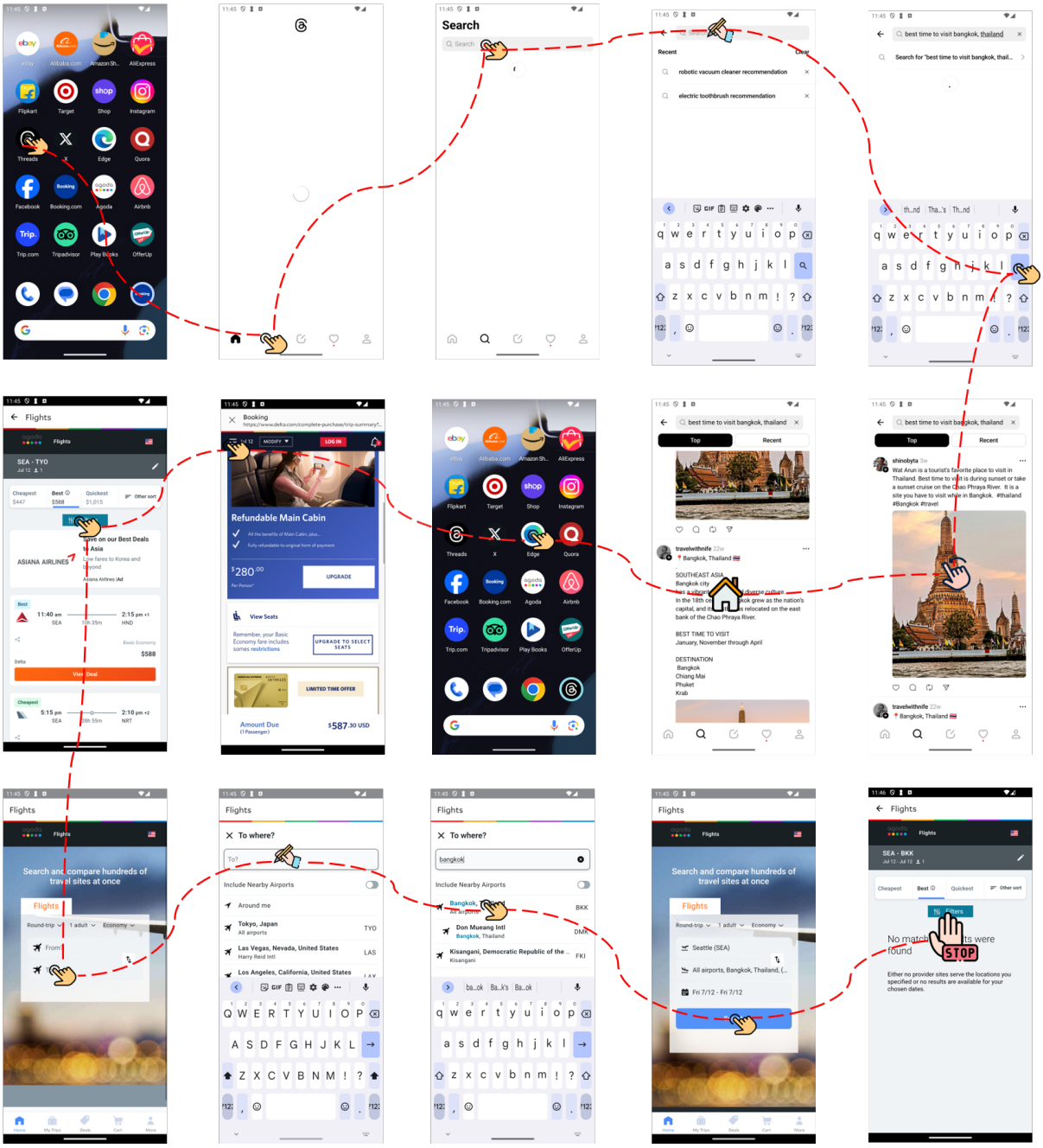

Figure 7: Schematic diagram of the GUI agent case study. The user provides an instruction to the agent, and the agent automatically executes it on a mobile phone or computer by simulating human operations.

Case study for agent tool-use domain

You are an expert in composing functions.
You are given a question and a set of possible functions.
Based on the question, you will need to make one or more function/tool calls to achieve the purpose.
If none of the function can be used, point it out.
If the given question lacks the parameters required by the function,also point it out.
Here is a list of functions in JSON format that you can invoke:

```
[
  {
    "name": "Holiday 100 Songs API",
    "description": "Provides information about the Greatest of All Time Holiday 100 Songs chart from
Billboard.",
    "parameters": {
      "type": "dict",
      "properties": {
        "year": {
          "description": "The year for which the chart information is required",
          "type": "int"
        },
        "artist": {
          "description": "The artist name for which the chart information is required (optional)",
          "type": ["string", "null"]
        }
      },
      "required": [ "year”]
    },
    "required": null
  },
  {
    "name": "Get Playlist Details",
    "description": "Retrieve details of a Spotify playlist, including playlist name, description, thumbnail, likes
count, tracks count, and details of each individual song.",
    "parameters": {
      "type": "dict",
      "properties": {
        "url": {
          "description": "The URL of the Spotify playlist",
          "type": "string"
        }
      },
      "required": ["url"]
    },
    "required": null
  },
]
```

Should you decide to return the function call(s).
Put it in the format of [func1(params_name=params_value, params_name2=params_value2...), func2(params)]
NO other text MUST be included.
Can you tell me which songs were on the top of the Billboard Holiday 100 chart in 2025?

Figure 8: Schematic diagram of tool-use agent. The user provides an instruction that requires the agent to complete the task by calling other tools, demanding the agent to correctly output the tool name, parameter names, and parameter values.