FEDERATED AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous AI Agents powered by LLMs have shown remarkable abilities in diverse domains. However, the training process typically require *centralized* collection of large amounts of real-world user data, posing substantial privacy and regulatory concerns. To this end, we explore a new *decentralized* training paradigm, namely **FEDAGENT** (Federated Agent Reinforcement Learning), which enables collaborative learning of AI agents across distributed clients without sharing local data. Moreover, we construct the first decentralized agent learning environment **FEDAGENTGYM**, which includes four types of LLM agents, two application scenarios (WebShop and ALFWorld), three variations of decentralized settings, and three newly defined heterogeneity challenges (*Preference Heterogeneity*, *Coverage Heterogeneity*, and *Hardness Heterogeneity*), to systematically investigate its effectiveness and impact factors. Extensive empirical studies show that FEDAGENT can have comparable performance to centralized agent training and exhibit strong robustness against heterogeneities, which shows the feasibility of training AI agents while protecting data privacy and opens a new direction for agent learning. The code is available here.

1 INTRODUCTION

The rapid advancement of AI agents, especially those powered by Large Language Models (LLMs), has demonstrated remarkable capabilities across diverse domains, from web navigation to embodied environments (Zhang et al., 2025; Gao et al., 2025; Liu et al., 2025). However, training these agents typically requires *centralized* access to vast amounts of users' real-world task query and trajectory data, which are inherently privacy-sensitive and hard to acquire due to regulatory compliance. Thus, a foundational question is: ***Can we train AI agents while protecting users' data privacy?***

In this paper, we explore a new *decentralized* training paradigm, namely **FEDAGENT** (**Federated Agent Reinforcement Learning**), which enables collaborative learning of AI agents, particularly LLMs, across distributed clients without sharing local data. In each round, the server distributes the current model to selected clients, who then train locally on their own data and send back their updated models. The server aggregates these updates by averaging them to create an improved global model for the next round. This process repeats iteratively, facilitating distributed LLM agent training while preserving data privacy since only model parameters, not raw data, are exchanged.

Compared with the previous federated learning literature, FEDAGENT is faced with fundamentally new challenges. The majority of existing federated learning research has concentrated on supervised classification tasks. There are also recent works that have explored federated reinforcement learning (FRL) for traditional RL settings (Qi et al., 2021; Kairouz et al., 2021; Liu et al., 2024). However, both of them operate under distinct assumptions compared to LLM agent learning. Supervised federated learning is usually built on static data distributions and one-shot predictions, while traditional FRL typically assumes simple rewards, low-dimensional state and action spaces. In contrast, LLM agent learning involves *natural language state and action spaces, diverse task formulations, and complex environment interactions*, which create entirely new challenges for federated paradigms. Thus, another essential research question is: ***Is FedAgent really effective for training LLM agents?***

To systematically investigate the effectiveness of this new training paradigm as well as the impact factors, we built the first decentralized agent learning environment **FEDAGENTGYM**, which incorporates four types of LLM agents from different model families (Qwen and Llama) and with different scales (1.5B, 3B, and 7B), two application domains (WebShop and ALFWorld), three dimensions of

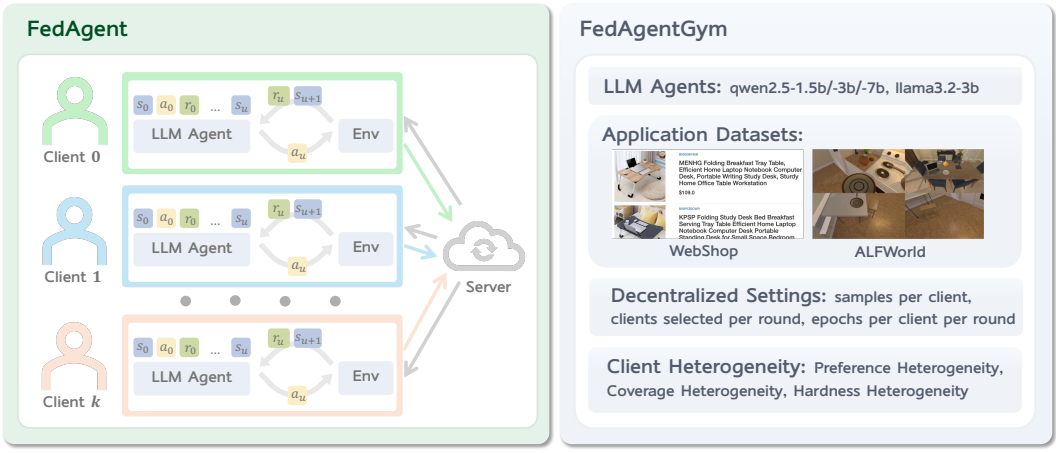

Figure 1: An Illustration of FEDAGENT and FEDAGENTGYM.

variation in decentralized settings (the number of samples per client, the number of clients selected per communication round, and the number of local training epochs per client per round).

Importantly, since the existing heterogeneity challenges in federated learning have mostly been defined in the context of supervised classification tasks (Gao et al., 2022; Ye et al., 2023), which focus on label skew, feature shift, or quantity imbalance, they are not directly applicable to LLM agent learning. Thus, we propose three new and orthogonal agent-specific heterogeneity definitions: ***Preference Heterogeneity***, where clients may prefer distinct types of tasks; ***Coverage Heterogeneity***, where the task sampling scope may vary across clients; ***Hardness Heterogeneity***, where the overall difficulty of tasks may differ among clients. Moreover, we carefully design three novel client partitioning strategies PREFERENCEPARTITION, COVERAGEPARTITION, and HARDNESSPARTITION accordingly, grounded in mathematical techniques such as *Gaussian Noise*, *Multinomial Sampling* and *Beta Distribution*. These strategies allow us to precisely control the extent of one type of heterogeneity across clients with a single hyperparameter, while keeping the other characteristics of the client distribution unchanged. We then incorporate them into FEDAGENTGYM to isolate and analyze the impact of each form of heterogeneity on FEDAGENT separately and controllably.

To investigate the effectiveness of FEDAGENT, we conduct an extensive and systematic study with FEDAGENTGYM. By training multiple LLM agents in the two application domains, we demonstrate that FEDAGENT consistently outperforms *local agent training* and can match the performance of *centralized agent training*, despite never sharing local data. By training LLM agents under different decentralized configurations, we discover the sensitivity patterns of FEDAGENT. Through our designed client partition strategies, we show FEDAGENT exhibits strong robustness to the aforementioned preference, coverage, and hardness heterogeneity challenges. Overall, our studies show the potential of scalable agent learning without sacrificing data privacy, provide valuable insights that inform practical deployment, and open new research directions for agent learning.

Our contributions can be summarized as follows:

- We made the first attempt to formulate and explore a new decentralized paradigm of training AI agents, namely **FEDAGENT** (**Federated Agent Reinforcement Learning**), which enables collaborative agent learning across distributed clients without sharing local data.
- We constructed the first decentralized agent learning environment **FEDAGENTGYM**, which includes four types of LLM agents, two applications (WebShop and ALFWorld), three variations of decentralized settings, and three heterogeneity challenges, to analyze the performance of FEDAGENT systematically and controllably, and offer insights to guide future development.
- We propose to categorize the new client heterogeneity challenges in decentralized agent learning into *Preference Heterogeneity*, *Coverage Heterogeneity*, and *Hardness Heterogeneity*. To investigate how each type of heterogeneity affects the performance, we introduce three novel client partitioning methods: PREFERENCEPARTITION, COVERAGEPARTITION, and HARDNESSPARTITION.
- Through extensive studies on FEDAGENTGYM, we provide three key insights on the effectiveness of FEDAGENT: (1) FEDAGENT not only outperforms single-client local training but also achieves

comparable performance to centralized agent learning. (2) FEDAGENT's effectiveness depends on decentralized configurations such as the number of clients selected per round and the number of epochs per client per round. (3) FEDAGENT shows high robustness against the aforementioned three types of agent-specific heterogeneity challenges.

- We release our code and environment as an extendable open-source library to inspire more future works in this new direction. The link to the repository is available here.

Algorithm 1 FEDAGENT with Client and Server training

Require: Total clients K, rounds T, clients-per-round M, local steps τ, learning rate η
Ensure: Final LLM-based global policy parameters θ_{final}
Initialize global policy parameters θ_0 (an LLM)
for $t = 0$ **to** $T-1$ **do**
  Server: sample client subset $S_t \subset [K]$ with $|S_t| = M$ (uniform without replacement)
  Server: broadcast θ_t to all $k \in S_t$
  for each $k \in S_t$ **in parallel do**
    Set local iterate $\theta_{k,t,0} \leftarrow \theta_t$
    for $i = 0$ **to** $\tau - 1$ **do**
      Collect a mini batch of trajectories $B_{k,t,i}$ using policy $\pi_{\theta_{k,t,i}}$ in environment $\mathcal{M}_k$
      Estimate policy gradient for $J_k(\theta_{k,t,i})$ on client k:

$$g_{k,t,i} \leftarrow \nabla_\theta \hat{J}_k(\theta_{k,t,i}; B_{k,t,i}) \quad (\textit{e.g.}, \text{GRPO})$$

      Local update: $\theta_{k,t,i+1} \leftarrow \theta_{k,t,i} + \eta\, g_{k,t,i}$
    end for
    Client returns local model $\theta_{k,t,\tau}$ ▷ equivalently $\Delta\theta_{k,t} = \theta_{k,t,\tau} - \theta_t$
  end for
  Server: Aggregation via model averaging:

$$\theta_{t+1} \leftarrow \frac{1}{M} \sum_{k \in S_t} \theta_{k,t,\tau} \quad \left(\text{equivalently } \theta_{t+1} = \theta_t + \tfrac{1}{M} \sum_{k \in S_t} (\theta_{k,t,\tau} - \theta_t)\right)$$

end for
return $\theta_{\text{final}} \leftarrow \theta_T$

2 FEDAGENT: FEDERATED AGENT REINFORCEMENT LEARNING

As shown in Algorithm 1, we consider a federated reinforcement learning setup for FEDAGENT. A population of clients are indexed by $k \in [K] = \{0, \ldots, K-1\}$. Training proceeds for communication rounds $t = 0, \ldots, T-1$. At round t, the server samples a subset $S_t \subset [K]$ of size $|S_t| = M$ uniformly without replacement, broadcasts the current global policy parameters θ_t, and aggregates the participating clients' locally updated parameters.

LLM Agent Training. The agent is a parametric policy π_θ (an LLM) that, conditioned on a task description c and an interaction history h_u up to step u, produces an action $a_u \sim \pi_\theta(\cdot \mid h_u, c)$. An action often contains both *a sequence of free-form tokens* (*i.e.*, the agent's intermediate reasoning) and *environment-facing choice* (*e.g.*, tool API calling). Each client k operates in a Markov Decision Process (MDP) environment $\mathcal{M}_k = (\mathcal{S}_k, \mathcal{A}_k, P_k, r_k, \rho_k, \gamma)$ with state space $\mathcal{S}_k$, action space $\mathcal{A}_k$, transition kernel P_k, reward function r_k, initial-state distribution ρ_k, and discount $\gamma \in (0, 1]$. Client k also has a distribution $\mathcal{D}_k$ over textual task descriptions $c \in \mathcal{C}_k$. Fix k and a task description $c \sim \mathcal{D}_k$. The agent interacts with $\mathcal{M}_k$ for horizon H, producing a trajectory:

$$\chi = (c, s_0, a_0, r_0, \ldots, s_H), \quad s_0 \sim \rho_k(\cdot \mid c),\ a_u \sim \pi_\theta(\cdot \mid h_u, c),\ s_{u+1} \sim P_k(\cdot \mid s_u, a_u, c).$$

The discounted return of χ is $R(\chi) = \sum_{u=0}^{H-1} \gamma^u r_u$. It is worth noting that when the LLM agents generate H consecutive textual actions $(a_0, ..., a_{(H-1)})$ in a trajectory χ, each action may span thousands of tokens, considering LLM agents' long reasoning capacity (DeepSeek-AI et al., 2025). This makes token-level credit assignment across the trajectory particularly challenging.

Local objective (client k). Client k aims to maximize the expected episodic return of the policy on its own environments and tasks:

$$J_k(\theta) = \mathbb{E}_{c\sim\mathcal{D}_k}\, \mathbb{E}_{\chi\sim(\pi_\theta,\mathcal{M}_k,c)}\big[R(\chi)\big]. \tag{1}$$

During round t, each participating client initializes a local iterate at the broadcast model, $\theta_{k,t,0} \leftarrow \theta_t$, and performs τ steps of stochastic policy optimization. At local step $i \in \{0,\dots,\tau-1\}$, the client collects a batch of trajectories $B_{k,t,i} = \{\chi^{(b)}\}_{b=1}^{N_{k,t,i}}$ by interacting with $\mathcal{M}_k$ under $\pi_{\theta_{k,t,i}}$ and computes a policy-gradient estimate:

$$g_{k,t,i} = \nabla_\theta \widehat{J}_k(\theta_{k,t,i}; B_{k,t,i}) = \frac{1}{N_{k,t,i}} \sum_{b=1}^{N_{k,t,i}} \sum_{u=0}^{H(\chi^{(b)})-1} \nabla_\theta \log \pi_{\theta_{k,t,i}}\big(a_u^{(b)} \mid h_u^{(b)}, c^{(b)}\big)\, \widehat{A}_u^{(b)}, \tag{2}$$

where $\widehat{A}_u^{(b)}$ is any valid return/advantage signal (*e.g.*, a GRPO-style estimator (Shao et al., 2024)). The local update is $\theta_{k,t,i+1} = \theta_{k,t,i} + \eta\, g_{k,t,i}, (i = 0,\dots,\tau-1)$ with step size $\eta > 0$. After τ steps the client returns its local model $\theta_{k,t,\tau}$ (equivalently the update $\Delta\theta_{k,t} = \theta_{k,t,\tau} - \theta_t$) to the server.

Global objective and aggregation. The federated learning goal is to maximize a weighted average of client objectives:

$$J_{\text{global}}(\theta) = \sum_{k=0}^{K-1} w_k\, J_k(\theta), \qquad w_k \geq 0, \;\; \sum_{k=0}^{K-1} w_k = 1. \tag{3}$$

In the FEDAGENT, the server uses uniform model averaging over the M participating clients each round (i.e., $w_k = \frac{1}{K}$ conceptually, with partial participation realized by S_t). Upon receiving the τ-step local models $\{\theta_{k,t,\tau}\}_{k\in S_t}$, the server performs model averaging: $\theta_{t+1} = \frac{1}{M}\sum_{k\in S_t} \theta_{k,t,\tau} = \theta_t + \frac{1}{M}\sum_{k\in S_t}\big(\theta_{k,t,\tau} - \theta_t\big)$. After T rounds the server outputs $\theta_{\text{final}} = \theta_T$.

3 FEDAGENTGYM: A DECENTRALIZED AGENT LEARNING ENVIRONMENT

3.1 LLM AGENTS AND APPLICATION DATASETS

FEDAGENTGYM is designed as an environment to investigate the impact factors of training AI agents, especially LLM, in a decentralized way. It includes four types of LLM agents, including `Qwen2.5-{1.5,3,7}B-Instruct` and `Llama-3.2-3B-Instruct`, and two challenging application datasets (WebShop (Yao et al., 2022) and ALFWorld (Shridhar et al., 2020)), which require complex reasoning process and multi-step environment interactions. We adopt these two datasets to simulate the real-world scenarios where data privacy concerns are paramount.

WebShop is a web-based interactive platform that evaluates LLM agents within authentic e-commerce scenarios. Task completion requires agents to navigate a simulated HTML shopping interface to locate, browse, and purchase appropriate items. The dataset features an extensive catalog of over 1.1 million products paired with $12,000$ user instructions, creating a rich and varied action space.

ALFWorld provides an embodied simulation benchmark that evaluates LLM agents' capacity for sequential decision-making tasks. Each scenario presents the agent with a textual objective that must be achieved through iterative environment interaction. The dataset encompasses $3,827$ task instances spanning six types of household activities: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick2).

3.2 DECENTRALIZED SETTINGS

We comprehensively examine the impact of different decentralized settings on FEDAGENT performance across three critical dimensions. First, we vary **the number of samples per client**, which determines the sampling scope for each LLM agent's exploration of the action space and response generation, directly affecting both the diversity of experiences collected and the quality of policy gradient estimates. Second, we change **the number of clients selected per communication round**, controlling both the computational parallelism and the degree of heterogeneity in exploration strategies incorporated during global model aggregation. Third, we adjust **the number of local training**

batches per client per round, governing the extent of local optimization on the sampled trajectories before synchronization with the central server. These parameters collectively influence fundamental trade-offs between exploration diversity, communication overhead, and convergence stability in the federated setting. Through extensive studies across these dimensions, we characterize how different decentralized training settings affect the final policy performance of FEDAGENT.

3.3 HETEROGENEITY CHALLENGES

To systematically evaluate how FEDAGENT performs under realistic client distributions, we propose three novel and orthogonal heterogeneity definitions, as conventional heterogeneity dimensions in federated classification tasks (*e.g.*, feature or label skew) (Gao et al., 2022; Ye et al., 2023) are not directly applicable. We also propose the corresponding client partitioning strategies, allowing us to understand the individual impact of different heterogeneity types.

Preference Heterogeneity: When Clients Have Different Task Preferences. In real-world federated learning, different clients often *prefer distinct types of tasks*. For example, in the ALFWorld, some users might frequently interact with kitchen-related tasks (like "put the apple in the fridge"), while others primarily encounter bedroom tasks (like "examine the lamp"). In WebShop, some may have mostly electronics searches while others mainly focus on clothing or home goods.

To simulate this preference heterogeneity, we propose the PREFERENCEPARTITION algorithm. The pseudo code is illustrated in Algorithm 2 in Section C.1. We model this by starting with the global distribution of task categories and introducing controlled noise to create client-specific preferences. Specifically, we add **Gaussian Noise** to the log-probabilities of the global category distribution, apply softmax normalization, and use the resulting probabilities to sample L instructions per client via **Multinomial Sampling**. As shown in Section C.2.1, this approach allows precise control over client distributions with a hyperparameter ω on topical preference heterogeneity, while maintaining the same total dataset size and per-client instruction count. More specifically, small noise values produce clients with similar task distributions, while larger noise creates highly specialized clients with distinct preferences.

Coverage Heterogeneity: When Clients Have Different Task Sampling Scopes. Even when clients encounter similar types of tasks, they may face vastly different quantities. A larger quantity of tasks indicates *coverage of a broader sampling scope per epoch* in reinforcement learning (we follow the setting in (Feng et al., 2025) to iteratively sample with replacement from the local data each epoch), while the sampling size remains fixed. Importantly, this differs from the quantity imbalance in conventional supervised federated classification tasks, where *training proceeds over the entire dataset each epoch*. In WebShop, for instance, some users might have extensive browsing histories with hundreds of product interactions, while others have only completed a few shopping sessions.

To model this heterogeneity, we develop the COVERAGEPARTITION algorithm. The pseudo code is shown in Algorithm 3 in Section C.1. We fix a global overlap target r (representing the average number of clients that see each instruction) and draw each client's data quantity from a **Beta Distribution**, which we then map to the range $[L_{\min}, L_{\max}]$. Task instructions are allocated to clients using weighted sampling without replacement to satisfy both individual client quotas and the global overlap constraint. As shown in Section C.2.2, this method isolates the effect of task sampling scope on FEDAGENT performance while keeping the underlying task distribution consistent across clients. Also, this method controls the extent of coverage heterogeneity via hyperparameter ξ without impacting the overall mean of client quantities.

Hardness Heterogeneity: When Clients Face Different Task Difficulties. A particularly important but often overlooked source of heterogeneity is the overall difficulty of tasks that different clients encounter, which can be quantified by the *success rate* of tasks. For example, in ALFWorld, some clients might consistently face simple navigation tasks with high success rates, while others encounter complex multi-step reasoning tasks that frequently result in failure.

As demonstrated in Algorithm 4 in Section C.1, our proposed HARDNESSPARTITION algorithm addresses this by partitioning the task instruction pool into "successful" and "unsuccessful" examples with a pretrained checkpoint. Then, using our COVERAGEPARTITION method, we first distribute successful instructions according to a **Beta Distribution** that determines each client's success rate. We then fill remaining slots with unsuccessful examples sampled uniformly, ensuring all clients have

exactly L instructions. As shown in Section C.2.3, this method enables us to study how different success rate distributions, which are controlled by a hyperparameter ξ' and measures the extent of hardness of task distributions for each client, affect FEDAGENT while maintaining consistent dataset sizes and global overlap patterns across all clients.

Method	**ALFWorld**							**WebShop**	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Qwen2.5-1.5B-Instruct									
Local (Client 21)	42.9	25.0	38.5	37.5	14.3	14.3	29.7	69.9	57.0
Local (Client 42)	50.0	37.5	**76.9**	25.0	42.9	14.3	45.3	75.1	53.1
Local (Client 84)	50.0	37.5	46.2	25.0	28.6	0.0	34.4	72.7	47.7
Centralized	$64.3_{\pm 4.8}$	$37.5_{\pm 0.9}$	$69.2_{\pm 6.1}$	$\mathbf{50.0}_{\pm 2.2}$	$42.9_{\pm 3.8}$	$28.6_{\pm 0.4}$	$51.6_{\pm 3.0}$	$79.9_{\pm 4.7}$	$57.8_{\pm 5.7}$
FEDAGENT	$\mathbf{80.0}_{\pm 4.2}$	$\mathbf{75.0}_{\pm 1.7}$	$53.8_{\pm 4.3}$	$37.5_{\pm 1.3}$	$\mathbf{83.3}_{\pm 4.7}$	$\mathbf{50.0}_{\pm 1.0}$	$\mathbf{64.1}_{\pm 2.8}$	$\mathbf{83.2}_{\pm 4.5}$	$\mathbf{61.7}_{\pm 1.8}$
Qwen2.5-3B-Instruct									
Local (Client 21)	41.5	12.5	34.9	**51.0**	18.9	21.2	31.3	59.8	55.0
Local (Client 42)	46.5	37.5	24.4	15.0	33.7	33.3	28.2	61.3	59.3
Local (Client 84)	22.8	27.5	39.1	46.3	48.3	36.5	29.9	77.6	58.6
Centralized	$94.1_{\pm 0.9}$	$\mathbf{80.0}_{\pm 2.5}$	$\mathbf{64.3}_{\pm 1.4}$	$42.9_{\pm 2.6}$	$50.0_{\pm 2.7}$	$22.2_{\pm 5.2}$	$62.5_{\pm 4.2}$	$\mathbf{86.0}_{\pm 1.5}$	$\mathbf{63.9}_{\pm 2.8}$
FEDAGENT	$\mathbf{95.5}_{\pm 4.3}$	$62.5_{\pm 3.0}$	$49.7_{\pm 1.7}$	$47.5_{\pm 2.4}$	$\mathbf{85.3}_{\pm 3.6}$	$\mathbf{45.1}_{\pm 2.1}$	$\mathbf{65.2}_{\pm 3.9}$	$85.5_{\pm 3.4}$	$63.1_{\pm 3.1}$
Qwen2.5-7B-Instruct									
Local (Client 21)	35.5	25.0	61.0	25.9	35.8	**45.2**	38.4	70.9	49.2
Local (Client 42)	29.0	45.0	18.8	25.6	15.9	38.0	42.1	85.2	33.6
Local (Client 84)	34.7	47.5	44.4	51.3	40.1	21.8	35.7	60.6	39.3
Centralized	$93.7_{\pm 4.5}$	$82.5_{\pm 2.1}$	$\mathbf{71.5}_{\pm 3.3}$	$47.9_{\pm 3.7}$	$63.2_{\pm 3.8}$	$31.9_{\pm 1.0}$	$73.3_{\pm 4.0}$	$78.8_{\pm 2.8}$	$64.7_{\pm 1.6}$
FEDAGENT	$\mathbf{94.5}_{\pm 2.3}$	$\mathbf{85.0}_{\pm 4.1}$	$56.0_{\pm 0.8}$	$\mathbf{62.5}_{\pm 1.2}$	$\mathbf{86.7}_{\pm 2.9}$	$42.8_{\pm 3.4}$	$\mathbf{75.5}_{\pm 2.9}$	$\mathbf{89.0}_{\pm 4.1}$	$\mathbf{68.9}_{\pm 3.8}$
Llama-3.2-3B-Instruct									
Local (Client 21)	39.8	50.0	17.9	40.0	20.7	34.0	38.1	65.3	50.5
Local (Client 42)	18.2	55.0	41.9	34.3	41.0	25.0	35.0	67.0	51.0
Local (Client 84)	29.9	32.5	39.0	18.9	18.8	**37.6**	29.7	70.2	55.7
Centralized	$72.4_{\pm 4.6}$	$\mathbf{62.5}_{\pm 4.5}$	$59.3_{\pm 3.1}$	$45.2_{\pm 0.5}$	$53.7_{\pm 2.2}$	$27.9_{\pm 3.0}$	$54.9_{\pm 2.9}$	$\mathbf{76.3}_{\pm 3.7}$	$56.2_{\pm 1.6}$
FEDAGENT	$\mathbf{83.7}_{\pm 1.7}$	$57.5_{\pm 6.0}$	$\mathbf{60.6}_{\pm 3.4}$	$\mathbf{55.9}_{\pm 0.9}$	$\mathbf{65.3}_{\pm 2.8}$	$24.9_{\pm 3.1}$	$\mathbf{61.2}_{\pm 3.3}$	$74.4_{\pm 4.9}$	$\mathbf{57.8}_{\pm 3.2}$

Table 1: **Performance Comparison on ALFWorld and WebShop**. We report the averaged performance and the corresponding standard deviation for Centralized Training and FEDAGENT over three random seeds. For ALFWorld, the **Success Rate** (%) is reported for each subtask as well as for the overall dataset. For WebShop, both the **Task Score** (%) and the **Success Rate** (%) are reported.

4 IS FEDAGENT COMPARABLE TO CENTRALIZED AGENT LEARNING?

Experiment Setup. In this section, we aim to investigate the performance of FEDAGENT under a uniform client distribution, which is independent of the aforementioned three types of client heterogeneities. We partitioned the whole dataset (WebShop, ALFWorld) into 100 clients. Each client has 100 task instructions and there is a potential overlap between clients. 2 clients are randomly selected each round. Each client is trained for 3 epochs per round, with a total of 70 rounds and 210 epochs overall. For each epoch, 64 tasks are sampled iteratively with replacement from local data.

As for FEDAGENT, we adopt GRPO (Shao et al., 2024) for policy optimization. Then, following the literature in federated learning (Liu et al., 2024), we select two typical baselines: **Centralized Agent Training** and **Local Agent Training**. Centralized Agent Training uses the full dataset (*i.e.*, 64 tasks are sampled iteratively from the whole dataset each epoch), while Local Agent Training uses only a specific client's dataset (we randomly selected one number 21 and obtained three client indexes 21, 42, and 84 as the baselines). Both of them run for the same total epochs as FEDAGENT and also adopt GRPO for policy optimization.

Result Analysis. As shown in Table 1, ❶ **FEDAGENT consistently outperforms Local Agent Training and achieves comparable performance to Centralized Agent Training**. For instance, on ALFWorld using `Qwen2.5-7B-Instruct`, FEDAGENT achieves a 75.5% total success rate compared to local training variants that range from 35.7% to 42.1%, while matching the centralized

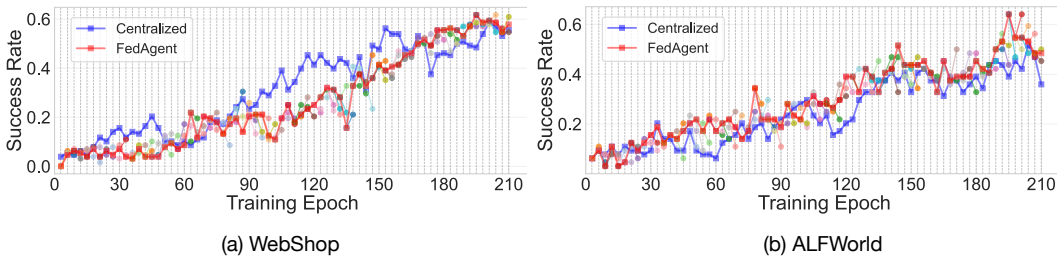

Figure 2: **Training Dynamics of FEDAGENT and Centralized Training**. Circle marks with different colors indicate the model performance after training on *specific selected clients each round*. The red line refers to the performance of the *aggregated models on server* throughout the training process.

training performance of 73.3%. This pattern is consistently observed across different model scales (1.5B, 3B, 7B), model architectures (`qwen` and `llama`), and randomly selected client indexes (21, 42, and 84). Similarly, on the WebShop benchmark, FEDAGENT maintains this advantage with `Llama-3.2-3B-Instruct` achieving 57.8% success rate versus local training with different indexes ranging from 50.5% to 55.7%, while remaining competitive with centralized training at 56.2%. These results demonstrate the advantage of FEDAGENT in achieving competitive performance while preserving users' data privacy inherently.

Figure 2 shows the whole training dynamics of FEDAGENT and Centralized Agent Training with `Qwen2.5-1.5B-Instruct` on WebShop and ALFWorld datasets. ❷ **Both paradigms ultimately converge to similar success rates despite different training dynamics** (~ 0.6 for WebShop, ~ 0.5 for ALFWorld). In WebShop (left), both approaches demonstrate steady monotonic improvement, with centralized training initially outperforming FEDAGENT until approximately epoch 120, after which both converge to similar success rates around 0.6. In contrast, ALFWorld (right) exhibits relatively more volatile training dynamics with frequent performance fluctuations for both methods, ultimately converging to success rates around 0.5. This further illustrates that FEDAGENT can achieve comparable performance with centralized training.

Insight 1 on the Effectiveness of FEDAGENT

FEDAGENT matches Centralized Agent Learning and outperforms Local Agent Learning: FEDAGENT incurs minimal performance penalty and benefits from diverse client contributions.

5 WHAT IS THE IMPACT OF DIFFERENT DECENTRALIZED SETTINGS?

Experiment Setup. In this section, we aim to study the impact of different decentralized settings on FEDAGENT in FEDAGENTGYM by systematically varying three key hyperparameters across two different datasets (WebShop and ALFWorld). We adopt `Qwen2.5-1.5B-Instruct` for all configurations. The experimental setup examines: (1) **samples per client**. We test 100, 500, and 1,000 tasks per client to understand how task sampling scope affects FEDAGENT learning dynamics; (2) **clients selected per round**. We compare 1, 2, and 4 participating clients each round to analyze the effect of federation scale on performance; and (3) **epochs per client per round**. We evaluate 1, 3, and 5 local training epochs to determine the optimal number of local computations before aggregation. Since we keep the total number of epochs the same at 210 for all configurations, 1, 3, and 5 local training epochs correspond to 210, 70, and 42 total rounds, respectively.

Result Analysis. The results in Figure 3 demonstrate that ❸ **FEDAGENT exhibits distinct sensitivity patterns towards decentralized settings**. First, it shows notable sensitivity to the number of epochs per client per round. Moving from 1 to 5 epochs per round leads to significant performance gains, especially after around 100 training epochs, highlighting that ❹ **shallow local updates may be insufficient to unlock the full potential of FEDAGENT**. On ALFWorld, FEDAGENT is also sensitive to the number of clients selected per round, with 2 clients per round outperforming 1 or 4, suggesting that ❺ **too few or too many clients could hinder convergence**. By contrast, FEDAGENT appears insensitive to the number of samples per client, as performance curves largely overlap across 100, 500, and 1,000 samples per round, suggesting that ❻ **the task sampling scope for one client**

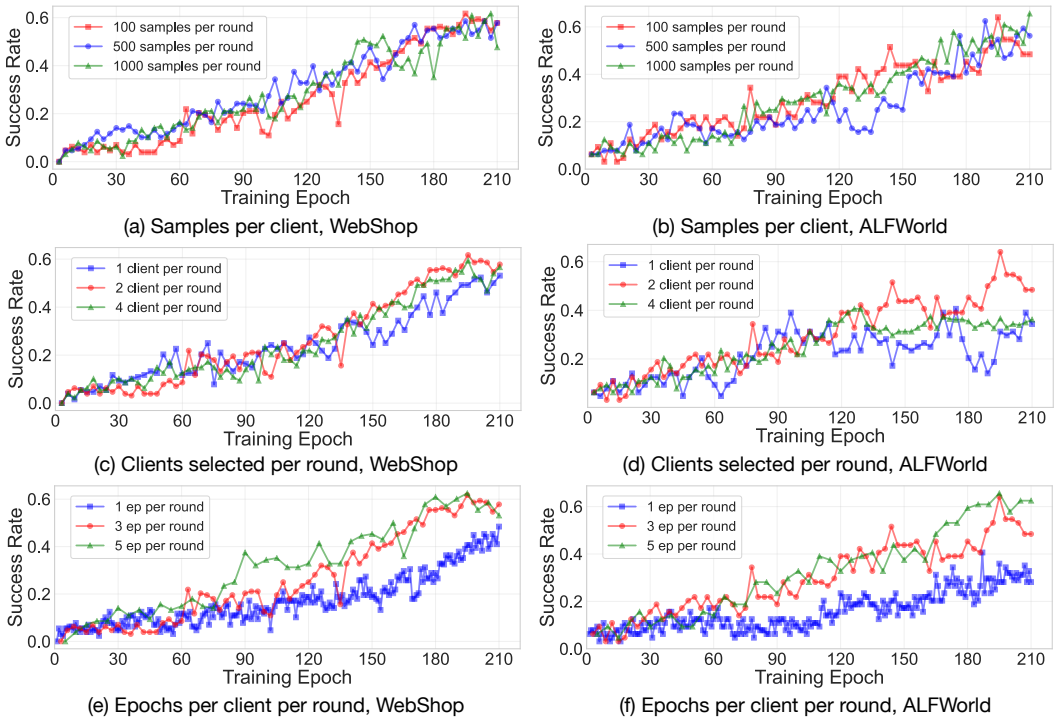

Figure 3: **Training Dynamics of FEDAGENT in Different Decentralized Settings**.

beyond a certain threshold may not be the limiting factor. Our studies offer valuable insights on the practical deployment of FEDAGENT and also suggest that ❼ **the specific decentralized configuration is important for FEDAGENT and may need to accommodate environments.**

Insight 2 on the Effectiveness of FEDAGENT

FEDAGENT's effectiveness depends on specific decentralized configurations: FEDAGENT exhibits relatively high sensitivity on clients selected per round and epochs per client per round, showing the importance of optimizing decentralized configurations.

6 WHAT IS THE IMPACT OF DIFFERENT HETEROGENEITY CHALLENGES?

Experiment Setup. In this section, we aim to study the impact of different heterogeneity challenges on FEDAGENT in FEDAGENTGYM. As shown in Appendix C.2, we can leverage our proposed client partitioning strategies PREFERENCEPARTITION, COVERAGEPARTITION, and HARDNESSPARTITION to precisely control the extent of one form of heterogeneity (Preference, Coverage, or Hardness Heterogeneity) across clients with a hyperparameter ω, ξ, or ξ', respectively, without affecting the others. We keep the number of total epochs as 210 and the number of all clients as 100, which are consistent with the main experiments. We adopt `Qwen2.5-1.5B-Instruct` in the experiments.

Result Analysis. As shown in Figure 4, ❽ **FEDAGENT shows high robustness against the three heterogeneity challenges**. Across all scenarios, preference heterogeneity (panels a,b), coverage heterogeneity (panels c,d), and hardness heterogeneity (panels e,f), even when comparing relatively uniform settings (blue lines in Figure 4, $\omega = 0.1$, $\xi = 256$, $\xi' = 256$) against extremely high heterogeneity settings (green lines in Figure 4, $\omega = 0.9$, $\xi = 1$, $\xi' = 1$), FEDAGENT consistently achieves strong success rates that steadily improve throughout training, The learning curves show that FEDAGENT maintains stable convergence behavior in both WebShop and ALFWorld environments regardless of heterogeneity intensity, with success rates generally reaching 0.5-0.6 by the end of training. Crucially, the performance degradation is minimal even under extreme heterogeneity conditions, indicating that ❾ **FEDAGENT has great potential to handle real-world scenarios across the full spectrum of heterogeneity challenges**.

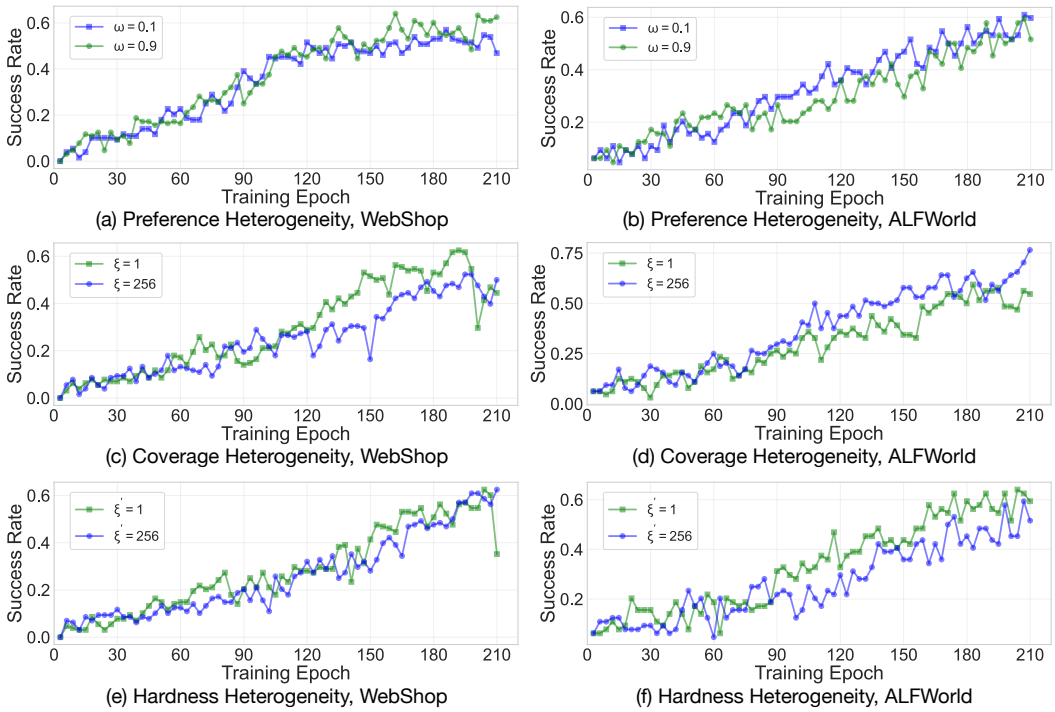

Figure 4: **Training Dynamics of FEDAGENT in Different Heterogeneity Challenges**.

Insight 3 on the Effectiveness of FEDAGENT

FEDAGENT shows high robustness against agent-specific heterogeneities: Under extreme conditions of Preference, Coverage, or Hardness Heterogeneity, FEDAGENT maintains stable convergence and achieves consistent final performance.

7 RELATED WORK

RL has been instrumental in empowering LLM agents to function effectively in dynamic and open-ended environments. Initial studies leveraged traditional RL approaches like DQN (Mnih et al., 2015) for training LLM agents in text-based gaming environments (Narasimhan et al., 2015), demonstrating the feasibility of using reinforcement learning to guide language-based decision-making in interactive settings. Subsequent research began incorporating value-based techniques across broader agent applications such as Android device manipulation (Rawles et al., 2023) and embodied environments like ALFWorld (Shridhar et al., 2020), showing that RL could scale beyond simple text games to more complex, multi-modal interaction scenarios. Contemporary methods have expanded RL training to encompass sophisticated web-based and application-specific tasks (Zhou et al., 2024; Putta et al., 2024), where agents must navigate complex user interfaces, execute multi-step workflows, and adapt to diverse task specifications. In previous works, real-world task queries and trajectories have been essential for training AI agents in practical applications. However, such data are becoming increasingly difficult to acquire due to privacy concerns. Our work makes an initial effort to explore training AI agents without compromising user data privacy, investigating alternative approaches that can achieve comparable performance while respecting data protection requirements.

8 CONCLUSION

In this work, we explored FEDAGENT (Federated Agent Reinforcement Learning), a new collaborative paradigm to train AI agent, particularly LLMs, across distributed clients, and built FEDAGENTGYM, the first decentralized agent learning environment. Extensive empirical studies demonstrate that FEDAGENT can achieve performance on par with centralized training and maintain strong robustness to heterogeneities. Our work validates the feasibility of training AI agents while protecting user data privacy and charts new research directions in agent learning.

REFERENCES

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv: 2501.12948*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

Dashan Gao, Xin Yao, and Qiang Yang. A survey on heterogeneous federated learning. *arXiv preprint arXiv:2210.04505*, 2022.

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

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*, 2025.

Bingyan Liu, Nuoyan Lv, Yuanchun Guo, and Yawen Li. Recent advances on federated learning: A systematic survey. *Neurocomputing*, 597:128019, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*, 2015.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. Federated reinforcement learning: Techniques, applications, and open challenges. *arXiv preprint arXiv:2108.11887*, 2021.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Mang Ye, Xiuwen Fang, Bo Du, Pong C Yuen, and Dacheng Tao. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44, 2023.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024.

Content of Appendix

A LLM Usage Disclosure **13**

B Reproducibility Statement **14**

C More Details of Heterogeneity Challenges **14**

C.1 Pseudo Code for Client Partitioning Strategies 14

C.2 Client Distributions under Partitioning Strategies 16

C.2.1 Preference Heterogeneity 16

C.2.2 Coverage Heterogeneity 17

C.2.3 Hardness Heterogeneity 18

A LLM USAGE DISCLOSURE

We hereby disclose that Large Language Models (LLMs) are utilized solely for the purposes of grammar correction and textual refinement.

B Reproducibility Statement

We performed all experiments on NVIDIA H100 GPUs, using the HuggingFace Transformers framework (https://huggingface.co/docs/transformers/en/index) as the basis for our implementation. To support further verification, replication and development, we release all the dataset, code and experimental outputs at the repository: https://anonymous.4open.science/r/federated_agent_submission-4652/README.md.

C More Details of Heterogeneity Challenges

C.1 Pseudo Code for Client Partitioning Strategies

Algorithm 2 PreferencePartition

Require: Category pools $\{\mathcal{I}_c\}_{c=1}^{C}$ with sizes n_c; total clients K; per-client set size L; jitter ω
Ensure: Client datasets $X_1, \ldots, X_K$ with $|X_k| = L$

$p_c \leftarrow n_c / \sum_{j=1}^{C} n_j; \quad \ell_c \leftarrow \log \frac{p_c}{1-p_c}$ ▷ global mix + logit anchors
for $k = 1$ to K **do**
  $z_c \sim \mathcal{N}(\ell_c, \omega^2)$ for $c = 1 \ldots C; \quad q_c \leftarrow \exp(z_c) / \sum_j \exp(z_j)$ ▷ larger $\omega \Rightarrow$ higher variance
  $(a_1, \ldots, a_C) \sim \mathrm{Multinomial}(L;\, q_1, \ldots, q_C)$ ▷ category counts for client k
  if any $a_c > n_c$ **then** set $a_c \leftarrow \min(a_c, n_c)$ and redistribute leftover by q to classes with spare capacity ▷ capacity fix within a set
  $X_k \leftarrow \bigcup_{c=1}^{C}$ SampleWithoutReplacement$(\mathcal{I}_c, a_c)$
end for
return $\{X_k\}_{k=1}^{K}$

Algorithm 3 CoveragePartition

Require: total items N (indexed $1{:}N$); total clients K; per-client bounds $(L_{\min}, L_{\text{avg}}, L_{\max})$ with $L_{\min} \le L_{\text{avg}} \le L_{\max}$; dispersion ξ; desired average replicas per item r
Ensure: Client datasets $X_1, \ldots, X_K$

$T \leftarrow \lfloor rN \rceil$ ▷ total assignments (sum of all $|X_k|$); keeps global overlap fixed
assert $KL_{\min} \le T \le KL_{\max}$ ▷ feasibility under per-client bounds
$\mu \leftarrow (L_{\text{avg}} - L_{\min})/(L_{\max} - L_{\min}); \quad \alpha \leftarrow \mu\xi,\ \beta \leftarrow (1-\mu)\xi$ ▷ Beta params with mean fixed at L_{avg}
Sample $x_k \sim \mathrm{Beta}(\alpha, \beta)$ for $k = 1 \ldots K$ ▷ larger $\xi \Rightarrow$ lower variance (sizes closer to L_{avg})
$u_k \leftarrow L_{\min} + x_k(L_{\max} - L_{\min}); \quad u_k \leftarrow u_k \cdot \dfrac{T}{\sum_j u_j}$ ▷ shape then renormalize to sum T
$n_k \leftarrow$ RoundToSum$(u,\, T,\, [L_{\min}, L_{\max}])$ ▷ largest remainder with clipping to $[L_{\min}, L_{\max}]$
$m \leftarrow \lfloor r \rfloor, \quad M \leftarrow \lceil r \rceil, \quad H \leftarrow T - mN$
Set $q_i \leftarrow M$ for any H items; $q_i \leftarrow m$ otherwise
Initialize $X_k \leftarrow \emptyset,\ \mathrm{rem}_k \leftarrow n_k$ for all k
for $i = 1$ to N **do** ▷ weighted, no-replacement placement across clients
  $\mathcal{A} \leftarrow \{k : \mathrm{rem}_k > 0\}$; choose q_i distinct $k \in \mathcal{A}$ with $\Pr(k) \propto \mathrm{rem}_k$
  Add item i to each chosen X_k and decrement the corresponding rem_k
end for
return $\{X_k\}_{k=1}^{K}$

Algorithm 4 HARDNESSPARTITION

Require: total items N (indexed $1{:}N$); disjoint index sets $\mathcal{S}$ (successful) and $\mathcal{U}$ (unsuccessful) with $\mathcal{S} \cup \mathcal{U} = \{1{:}N\}$; total clients K; per-client set size L; Hyperparameters for COVERAGEPARTITION: bounds (ℓ, c, h) with $h \leq L$, dispersion ξ', overlap r

Ensure: client datasets $X_1, \ldots, X_K$ with $|X_k| = L$

$\{Y_k\}_{k=1}^{K} \leftarrow$ COVERAGEPARTITION$\big(|\mathcal{S}|, K, (\ell, c, h), \xi', r\big)$ ▷ larger $\xi' \Rightarrow$ lower variance
for $k = 1$ to K **do**
  $m_k \leftarrow L - |Y_k|$; $F_k \leftarrow$ SAMPLEWITHOUTREPLACEMENT$(\mathcal{U}, m_k)$
  $X_k \leftarrow Y_k \cup F_k$
end for
return $\{X_k\}_{k=1}^{K}$

C.2 Client Distributions under Partitioning Strategies

C.2.1 Preference Heterogeneity

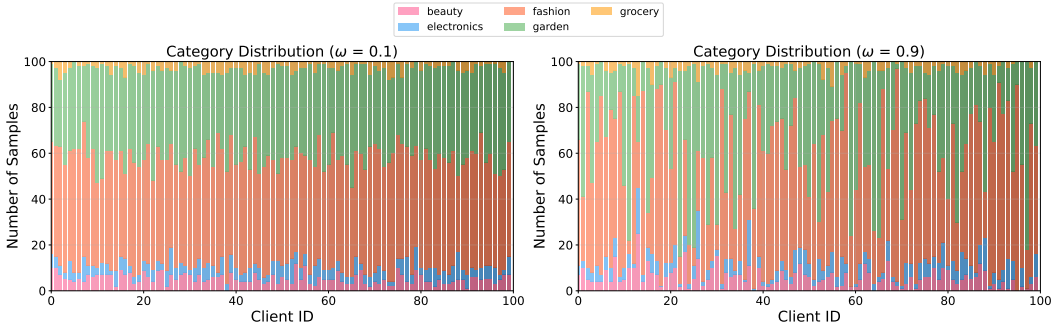

Figure 5: **Client Distribution under Preference Heterogeneity (WebShop,** $\omega = 0.1$ **vs.** $\omega = 0.9$**).**

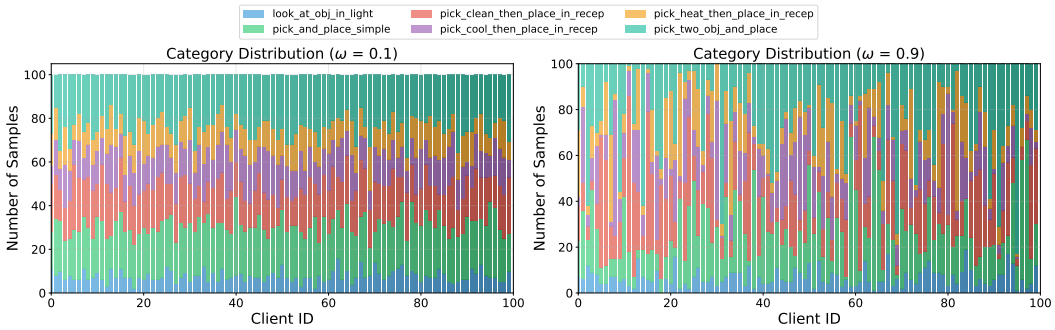

Figure 6: **Client Distribution under Preference Heterogeneity (ALFWorld,** $\omega = 0.1$ **vs.** $\omega = 0.9$**).**

C.2.2 COVERAGE HETEROGENEITY

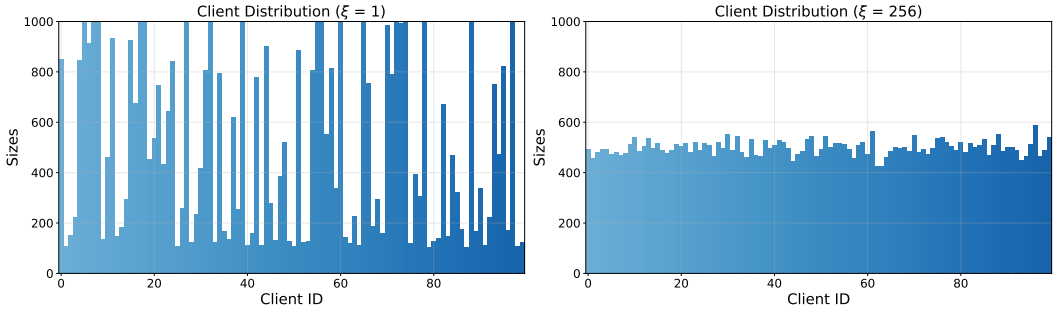

Figure 7: **Client Distribution under Coverage Heterogeneity (WebShop, $\xi = 1$ vs. $\xi = 256$).**

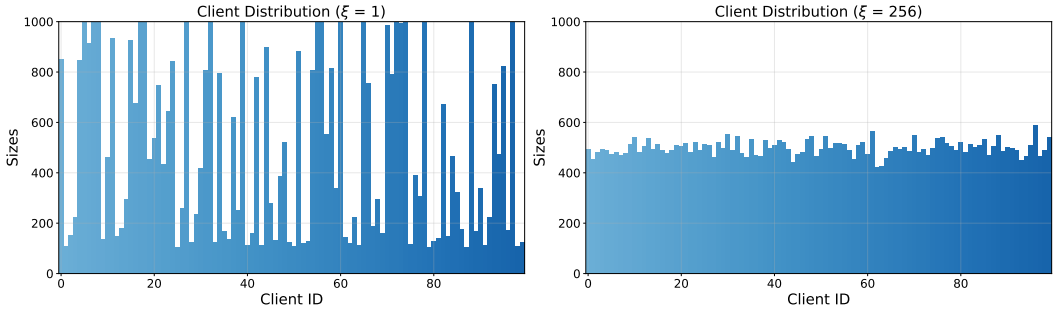

Figure 8: **Client Distribution under Coverage Heterogeneity (ALFWorld, $\xi = 1$ vs. $\xi = 256$).**

C.2.3 HARDNESS HETEROGENEITY

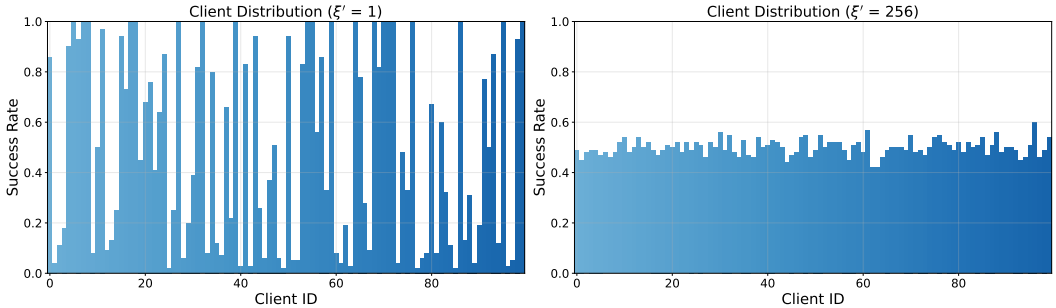

Figure 9: **Client Distribution under Hardness Heterogeneity (WebShop, $\xi' = 1$ vs. $\xi' = 256$).**

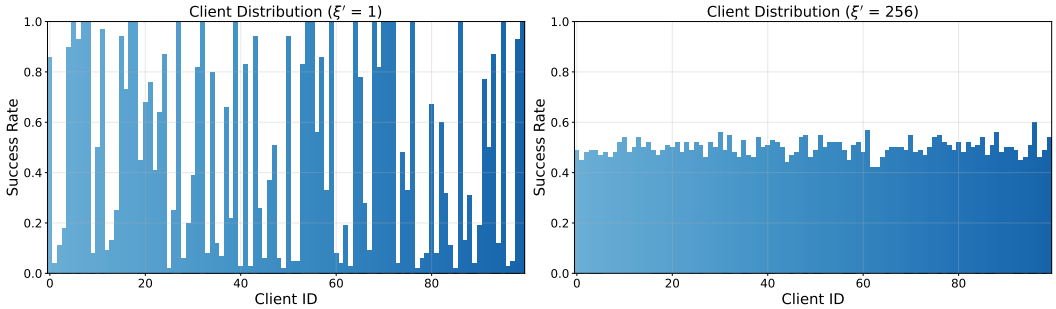

Figure 10: **Client Distribution under Hardness Heterogeneity (ALFWorld, $\xi' = 1$ vs. $\xi' = 256$).**